The Design and Simulation of an Inverter

Cadence Tutorial

This tutorial has been devised to run through all the steps involved in the design and simulation of a CMOS inverter using the Cadence CAD tools. Before invoking the Cadence tools it is essential to first set up the environment under which these tools will be used throughout the semester.

Setup

All of the tools we use will be run in Linux on the CAEN lab machines and login servers. They can also be run on EECS departmental research machines that have paid for software access and been set up properly. It is also possible to run them (slowly) remotely from linux, windows, and macs client machines at home. Setup instructions for remote access from these systems in appendix C of this document. When sitting down at a CAEN lab PC you should:

1. Reboot into linux if it is in windows. When you reboot a CAEN lab machine a boot OS menu should pop up for a few seconds. It is easy to miss it. Note that there are generally two consecutive boot menus (one is part of windows and one is part of Linux) This is expected even if it is annoying.
2. Log in with your uniqname and CAEN password (usually your Kerberos password but this can get messed up)
3. Open a command line terminal (right click on the wallpaper and choose terminal). All of the instructions in this tutorial will will be given for the command line or for the specific gui applications you start from the command line. In this tutorial the % symbol indicates the UNIX prompt of the c-shell (so you shouldn't type it in) and is usually preceded by the name of the machine you are working on.

First, open your .cshrc file, which gets read every time you open a terminal (actually it is slightly more complicated than that but the explanation suffices for our purposes). Do this using the text editor of your choice (nedit, nano, gvim, vi, emacs, etc.) We have experience with any of those listed except for emacs.

```
% nedit ~/.cshrc
```

Add the following lines to the bottom of the file:

```
if ($?prompt) then
  if (! -f /afs/umich.edu/class/eecs427/tsmc25/cshrc) then
   gettokens
  endif
  source /afs/umich.edu/class/eecs427/tsmc25/cshrc
endif
```

Save. Now close the terminal and open a new one. You may be asked for your password again. This is from the gettokens command in your .cshrc which is there because of the current setup of AFS at CAEN. If your home directory is in the engin.umich.edu cell (should be true for most people who have been here a few years) you do not get umich.edu tokens by default when you log in. You can ask CAEN to change your home directory to the umich.edu cell to eliminate this annoyance (they will "forcibly" move everyone soon anyway).

For the remainder of the course you will be working in your EECS 427 class directory which will store all the files that you create. We will create a symbolic link in your home directory to your class directory as so:

```
% ln –s /afs/umich.edu/class/eecs427/w07/<YOUR_UNIQUENAME> ~/eecs427
```

The last global setup step is to delete any existing .cdsenv file that might mess things up with this command:

```
% rm ~/.cdsenv
```

Overview of Full-custom Design Flow

The following steps are involved in the design and simulation of a CMOS inverter.

1. Capture the *schematic* i.e. the circuit representation of the inverter. This is done using the Cadence tool *Composer*.

2. Create a *symbol*. The design will be needed in higher schematics including a testing schematic and hence it needs to be represented by a *symbol*. This too is done using *Composer*.

3. Verify correct logic functionality using the verilog simulator *NC Verilog*. This digital simulation is not as accurate as analog simulation but is much faster so more complex logical testing can be done.

4. Create an analog testing schematic, also with *Composer*.

5. Verify correct analog functionality and get initial timing information with the spice simulator *HSpice* (actually a Synopsys product rather than a Cadence product) to further guarantee correct operation prior to beginning full-custom layout. This step isn't always necessary in the design flow but can help to size transistors and is more accurate than verilog simulation.

6. Physically *lay out* the inverter according to some CMOS process rules. In our case we will be using the TSMC 0.25 micron CMOS process with modified MOSIS SCMOS DEEP SUBM design rules available as a separate handout. Layout is done using the *Virtuoso Layout Editor*. Note that dimensions in *VLE* for this kit are in microns so you will need to multiply lambda values by .12 microns to get the exact size to draw in *VLE*.

7. Check the layout to verify that it conforms to the process design rules. This is done using *Diva DRC*, which is accessible from within *Virtuoso*.

8. Check that the layout and schematic match using *Diva LVS*, a layout versus schematic checker that does this and is also available from within *Virtuoso*.

9. Extract parasitic layout parameters such as capacitances which contribute to circuit delays. This is also done using *Diva* from within *Virtuoso*.

10. Simulate with *HSpice* once again to determine circuit delays and verify correct operation in the presence of these parasitics.

11. Annotate the schematic to reflect the circuit delays you calculate in *HSpice*.

12. Simulate with *NC Verilog* again with the new digital circuit delays.

Now, we are ready to begin the design. Making sure you are in your class space, make a cad1 directory to store your work and set up the cadence initialization files in it. We are naming it cad1 because the inverter we will work on here is part of your first cad assignment.

```
% cd ~/eecs427
% mkdir cad1
% cp $CDK_DIR/cds.lib $CDK_DIR/.cdsinit cad1
% cd cad1
```

Cadence Tools

In this course most of the work will be done from within the Cadence Front to Back environment. To invoke this environment perform the following command at the prompt.

```
% icfb &
```

The & causes icfb to be run in the background so that you can continue to use your terminal for other things.

Once it starts the first window that appears is called the *CIW* (Command Interpreter Window). The other window that appears is the *Library Manager*. The Library Manager allows you to browse the available libraries and create your own.

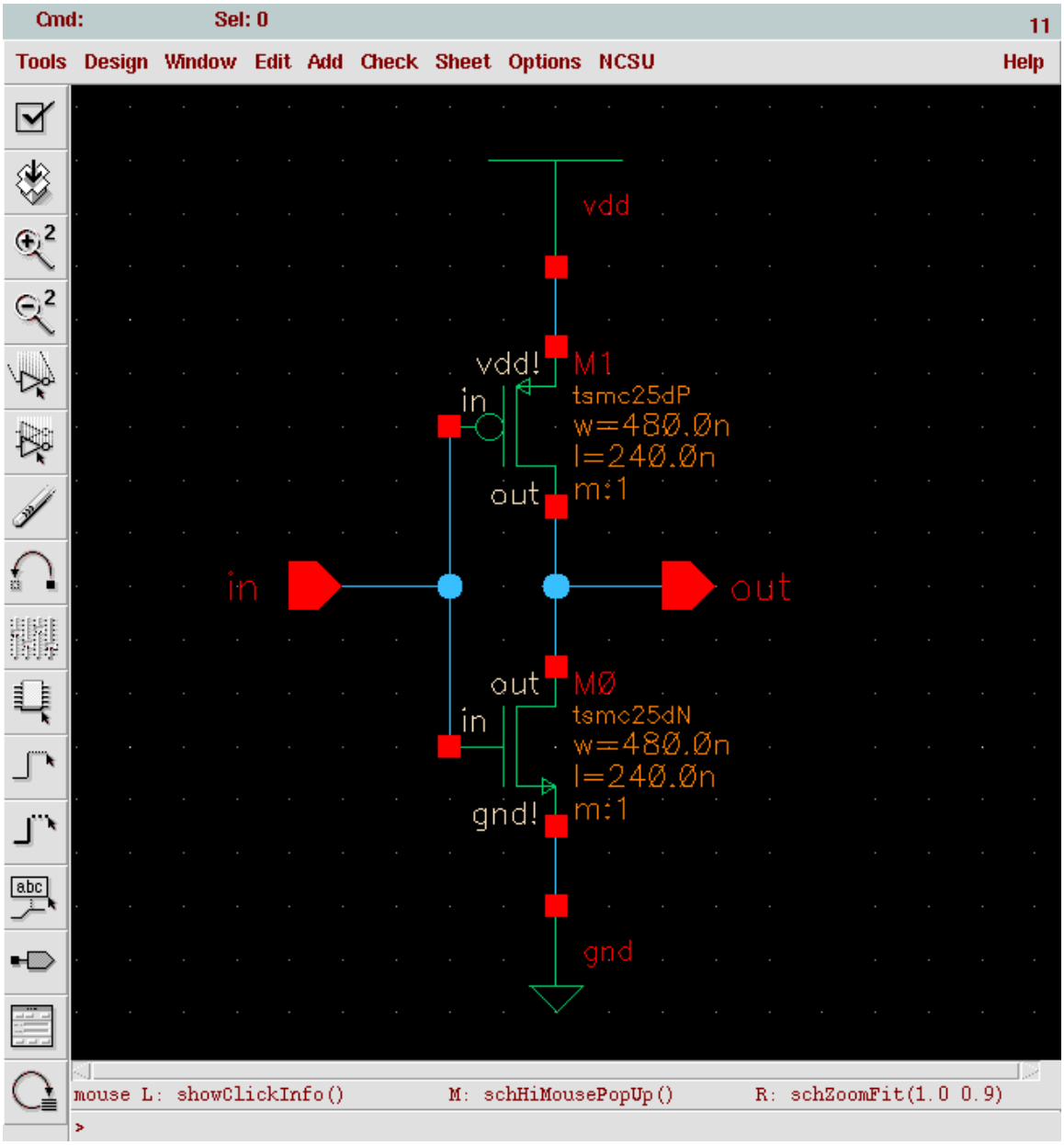
Create the Inverter Schematic

In the Library Manager, create new library called inverter. Select ***File->New->Library***. This will open new dialog window, in which you need to enter the name of your library and library location. Enter “cad1” for the name (don’t change the location) then press ok. You will be prompted and you should choose "Attach to existing tech library". You should set the Technology Library to “NCSU_TechLib_tsmc03d” via the pulldown menu and click OK.

You should now see the library "cad1" appear in the Library Manager.

Next, select the library you just created in the Library Manager and select ***File->New->Cell View...***. We will create a schematic view of an inverter cell. Type in "inverter" under cell-name and "schematic" under view. Click OK. Note that the "Tool" is automatically set to "Composer-Schematic", the schematic editor. Alternatively, you can select the "Composer-Schematic" tool, instead of typing out the view name. This will automatically set the view name to "schematic".

After you click "OK", the blank Composer screen will appear. The image below shows the final schematic that we will make in this tutorial.



Inverter Schematic

To generate a schematic like this, you will need to go through the following steps:

From the Schematic Window, choose ***Add->instance***. The *Component Browser*, will then pop up. In the Library field, select NCSU_Analog_Parts. We will place the following instances in the Schematic Window from the NCSU_Analog_Parts library as instructed below:

N_Transistor:	nmos
P_Transistor :	pmos
Supply_Nets :	vdd , gnd

Place pmos instance **-** In Component Browser, select *P_Transistors* and then *pmos*. Place it in the Schematic Window.

Do the same for the nmos, gnd, and vdd. (gnd and vdd can be found in Supply Nets)

Place IN pin **-** From the Schematic Window menu, select *Add->Pin*... In the Pin Name field , enter *in*. In the Direction field, select *input*. Place it in the Schematic Window.

Place OUT pin **-** From the Schematic Window menu, select *Add->Pin*.... In the Pin Name field , enter *out*. In the Direction field, select *output*. Place it in the Schematic Window**.**

Place wires **-** In the Schematic Window menu, select *Add->Wire (narrow)* (or simply press the 'w' key). Place the wire to connect all the instances. Be sure to click once for the start point and once at the endpoint, do not click and drag to draw the wire.

Save Your Design - Select *Design->Check and Save*.

Look at the CIW. You should see a message that says:

Schematic check completed with no errors.
"cad1 inverter schematic" saved

If you do have some errors or warnings the CIW will give a short explanation of what those errors are. Errors will also be marked on the schematic with a yellow or white box. Errors must be fixed for your circuit to simulate properly. It is up to you to decide if you should fix warnings of not. The most common warnings occur when there is a floating node or when there are wires that cross but are not connected. Just be sure that you know what effect each of these warnings will have on your circuit when you simulate.

Your schematic should look like the one shown above. Note that your transistors will have the model **tsmc25dN** and **tsmc25dP**, the widths will be **360n**, and the lengths will be **240n**. We will want to change these parameters. First escape out of whatever mode the schematic editor is in by pressing escape. Now select the device you wish to change the properties for. You can bring up the properties menu by one of three methods:

1) Click the right mouse button, select Properties from the pop-up menu
2) Select the Properties Icon on the left side of the Schematic Window
3) Press the 'q' key on the keyboard

Now change both the nmos and pmos instances to have width **480n** and length **240n**. And check and save your design again.

Create the Symbol

In order to use this symbol in higher level schematics we need to generate a symbol for it. To do this click *Design->Create Cellview->From Cellview*. In the dialog that appears make sure "To View Name" is set to "symbol", and that the "tool" is "Composer-Symbol" and press OK. A new editor window should be generated with the symbol, you should only need to regenerate this symbol if the input or output pins change on your schematic. Check and Save the symbol. If you look in the Library Manager window you should now see both the schematic and the symbol in cell views.

Digital Logic Simulation

We will now simulate the schematic design using NC Verilog. This will be done to verify the correct digital behavior of the schematic. The first step is to generate the verilog netlist and simulation infrastructure from the schematic. Open the schematic view of the inverter if it is not already open. Click *Tools->Simulation->NC-Verilog* from the menu. A new window will appear. Pick a run directory name, this time it is ok to leave it inverter_run1. Click the top icon (of a person running) to initialize the design, this will create the necessary files in the run directory. After the design is initialized click on the next icon down (of three check boxes) to generate the netlist. This will take your schematic and make a verilog file out of it to be simulated.

Now before you click the third icon down to run the verilog simulation you need a testbench to exercise the inputs. The tools created a very stupid testbench for you in the file testfixtures.verilog within the inverter_run1 directory for you that sets all of the inputs to 0 and then quits. The idea is that you will modify this to be more meaningful. Open this with your favorite text editor from the command line so that we can set up the values we want to simulate on the inputs to our design. You should be in your cad1 directory already so

```
% cd inverter_run1
% nedit testfixtures.verilog
```

This is part of a simple verilog module that will stimulate the inverter. The actual test module is located in the testfixtures.template file and refers to this file for the meat of the testbench. This will be a very simple testbench but you can use this as a template for creating more complex input stimuli for other designs, and the tools have done a lot of the busy work for you that newcomers to verilog can easily screw up.

The initial block provides the stimulus for the inverter test. The inverter input is initially set to the logic value 0 (1'b0). We will want to simulate more than this so we will begin by modifying the initial block to look like the following:

```
initial
begin
   in = 1'b0;
   #10 in = 1'b1;
   #10 in = 1'b0;
   #10 $stop;
end
```

This input will now toggle the value of in after 10ns (#10), then again 10ns after that. In this particular initial block, there is a $stop (10ns after the second toggle). This will stop the simulation. It's important to include a $stop statement, otherwise the simulator can continue to run indefinitely and fill up a disk volume with probed output signal data. Initial blocks are executed once per simulation and are commonly used in generating input stimulus. Initial blocks begin execution at time=0. The sequencing of events is determined by the delay (#) annotation. Another useful construct to understand is the *always* block. An *always* block is executed continuously throughout the simulation. Since it continuously executes, you must incorporate time control in the block otherwise you can create an infinite loop. We'll talk more about the *always* block later in EECS 427 but for now we're really only interested in using it to generate periodic input stimulus. An example of an always block that generates a 50% duty cycle clock with a period of 100ns is provided below. It triggers continuously after every 50ns. Following this always block is an initial block that will set the clk signal at time=0 (otherwise it will remain unknown). The "#50" is important. If you remove it the always block will trigger every 0ns and the simulator will exit reporting an oscillation problem.

```
always
begin
  #50 clk = ~clk;
end

initial
begin
  clk = 1'b0;
end
```

Finally, look at how logic values have been specified. Logic 0 was written as 1'b0 and logic 1 was written as 1'b1. We could have written just 1 or 0 but expressing them this way gives insight into how you might drive, say, a 16-bit data bus with an hex-encoded opcode of 1AF4. For this definitions of databus:

```
reg databus[15:0];
```

Here's how you would do that:

```
databus = 16'h1af4;
```

Other ways of driving databus are:

```
databus = 16'b0001101011110100;
databus = 16'd6900;
databus[15:8]=8'h1a; databus[7:0]=8'hf4
databus[15]=1'b0; databus[14:0]=15'h1af4
```

The first field of the number declares the bit-width, followed by a "'", then by an identifier: h - hex, d - decimal, o - octal and b - binary and finally by the number in the given encoding. The hex and binary representations will be the most useful. Also note how the reg signal databus was declared 16-bits wide with the "[15:0]" and then slices of the bus were refenced with the same notation. **Please learn bus notation and use it often**. It will make your lives easier when you move on to bigger projects during the semester.

You now have a complete verilog model for your inverter along with a verilog module that instantiates and stimulates the inverter (remember to save the tesbench). At this point you're ready to bring up NC Verilog to simulate your design by clicking the third icon down (of a box over a square wave) in the icfb NC Verilog interface.

When NC Verilog starts, SimVision will open. There are several windows that we could open, but for now we will just concern ourselves with the Waveform viewer and the Browser. The browser should already be open and you should see the simulation environment in the left pane. It is comprised of the top level "simulator" which represents everything to be simulated and the two sub categories "cds_globals" and "test". The cds_globals files present the simulator with the logic values for vdd and gnd. The test folder represents the test environment we have set up. Click on the "test" folder to see what it contains.

If you look in the middle pane, you can see the two signals in the test module. They are in and out. We will want to add these to the waveforms in a minute, but for now expand the test module by clicking the plus sign beside it, you will see that test is composed of a module called top, which also contains the signals in and out. Notice that in the module top, in and out are actual inputs and outputs, so the icon includes an arrow to signify this. Continue to expand the hierarchy and explore, you should be able to find the instances of the transistors (M1 and M2), and see they have source, gate, and drain ports (S,G,D).

Now that you have learned to explore the design in the browser, lets go back to the test level by clicking on test. Lets start the waveform window by adding in and out to it. This can be done by either selecting the signal and pressing the *Send to Waveform* button, or by right clicking and choosing *Send to Waveform*.

We can simulate our testbench by selecting *Simulation->Run*. If you modify the testbench while simvision is open you can get the new testbench with *Simulation->Reinvoke Simulator.* For the first few assignments this will be sufficient. As the assignments become more complex you may want to explore the options of stepping the code, setting breakpoints, and viewing the source. For more information on these you can reference the SimVision Waveform user guide on the course webpage.

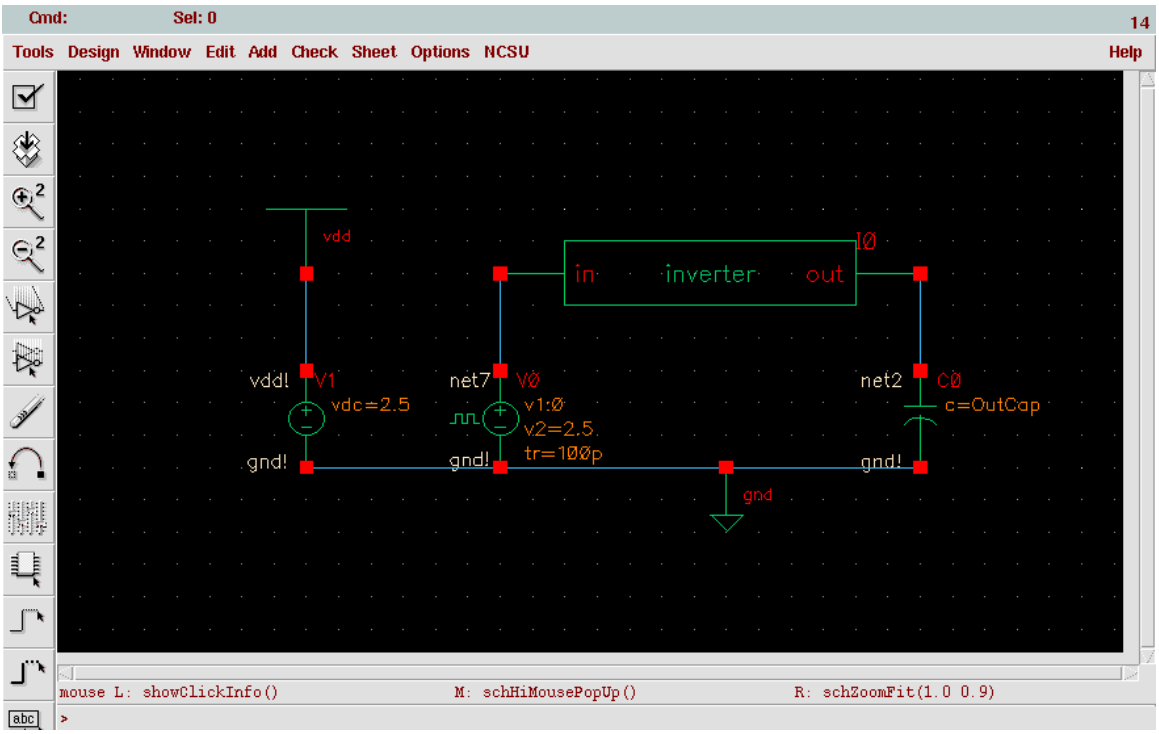
On the waveform viewer you should now see 30ns of simulated time, with the input and output changing at 10ns and 20ns. Verify that the correct logical operation is being performed (inversion). For the first CAD assignments it will be sufficient to turn in a printout of the proper logic simulation. To do this select, *File->Print Window* in the waveform window. In the dialog that appears, change *Command* to be *Print to File.* Enter the name of the file to print with a .ps extension. You can add details to the printout in the fields on the form as well. In later CAD assignments you will need to generate a database for submission, because viewing proper functionality would be too small on a single printed page for complex circuits like an ALU. When this is needed the CAD assignment will provide directions on how to do this.

You can close all of the ncverilog and simvision windows for now.

Analog Circuit Simulation

In order to run an analog spice simulation of your circuit you will need the analog equivalent of a testbench. In this flow that is another schematic that uses your part as a black box.

In the Library manager select your cad1 library and select *File->New->Cell View*.... We will create a schematic view of an inverter testing circuit. Type in "inverter_test" under cell-name and "schematic" under view. Click OK. After you click "OK", the blank Composer screen will appear. The image below shows the final schematic that we will make in this tutorial.



Inverter test schematic

Placing the vdd, gnd, and wires should be familiar from the schematic of the inverter itself. The inverter inverter symbol is from your cad1 library, and the other three symbols to place from the NCSU_Analog_Parts library are vdc and vpulse from Voltage_Sources and cap from R_L_C. Note that these parts pop up a dialog box to enter properties before placing. If you mess up the properties you can always go back and edit them the same way you edited the nmos and pmos properties in the inverter schematic.

For the vdc instance : In the DC voltage field, enter 2.5. V should be filled in automatically

For the cap instance : In the Capacitance field enter OutCap F should be entered automatically. (This Design Variable will be used later in the Analog Environment allowing us to easily modify it.)

For the vpulse instance enter the following values in the form. Appropriate units of V and s should be added automatically.

Voltage 1:	0
Volrage 2:	2.5
Delay Time:	0
Rise Time:	100p
Fall Time:	100p
Pulse Width:	900p
Period:	2n

Now check and save your schematic.

We placed a VDC component to specify what our VDD value is 2.5v as well as an output capacitor to represent circuit load to our circuit. We also placed a vpulse on the input signal. We set the parameters for the pulse to be a square wave with a frequency of 2ns and a rise and fall time of 100ps. There are several other useful voltage sources in the library, particularly the VPWL, which allows you to specify a piece-wise linear voltage input.

You are now prepared to simulate your circuit. From the Schematic Window menu, select *Tools->Analog Environment* A window will pop-up. This window is the Analog Artist Simulation Window.

We will first need to setup the simulation type. From the Analog Artist menu, select Setup -> Simulator/Directory/Host. Choose hspiceS as your simulator. Your simulation will run in the specified Project Directory. You may choose any valid pathname and filename that you like.

Next you will set up the analysis. We will do Transient Analysis on the circuit that we just produced. From the Analog Artist menu, select *Analyses->Choose*... Fill out the form with the following values: From: 0 To: 15n By: 10p

Next we will add the variable from our schematic (OutCap). From the Analog Artist menu, select *Variables->Edit* The Editing Design Variables form will appear. Fill out the form with OutCap as the name and 0.01p as the value, and then click Add to send this Variable to the Table of Design Variables. (Recall that we entered the OutCap Design Variable in the Capacitor component while editing the schematic earlier.)

Now we are ready to run the simulation. From the Analog Artist menu, select *Simulation->Run*, Look at the echoing information in the CIW window. If the simulation succeeds, the window will display "...successful."

If the simulation is unsuccessful, then one of the error messages should provide a clue as to what went wrong. Remember that you can move elements around in your schematic by clicking and dragging them. You can delete them by selecting them and pressing the "delete" key. You modify the properties of the elements by selecting them and pressing the "q" key.

Once we have run the simulation we can view the waveforms. From the Analog Artist menu, select *Results->Direct Plot->Transient Signal* . Now we are prompted to enter the signals we want plotted. In the Schematic Window, Click on the IN wire, Click on the OUT wire, and then Press ESC on your keyboard. The waveform window will now open. The two curves (IN and OUT) will then be displayed in this window: To separate them on their own axis, select *Axis->Strips.*

You can zoom in on a region by drawing a box with the mouse. For now you will want to focus on either a falling or rising transition of the output. Zoom in on one of these. Remember that you can also use the menu to return to a graph with everything (fit). Try placing a couple of vertical markers at the 50% (.9v) of both the input and output waves. You do this by selecting *Marker->Place->Vert Marker* and then clicking on a point on the graph. After you have placed the markers try moving them around. This is difficult to do at first, but click on the label (time/voltage) and the marker should be highlighted. Then click on the point where the marker crosses the curve and drag it along. Try to get the markers as close to the 50% voltages as possible. You can now calculate the delay time by subtracting the two times from the markers. To find the rise/fall times you would just put markers at 10% and 90% of the output transition and measure the time change. There are more sophisticated things that you can do in the waveform, but for now this is sufficient to find the times we ask for.

You should generate one graph showing each of the timing information we ask. (One for rise time, one for fall time, one for rise delay, and one for fall delay). You can save the graph to a file by clicking *File->Save As Image.* It will prompt you for a type (we prefer png) and name for the .png file it will create.

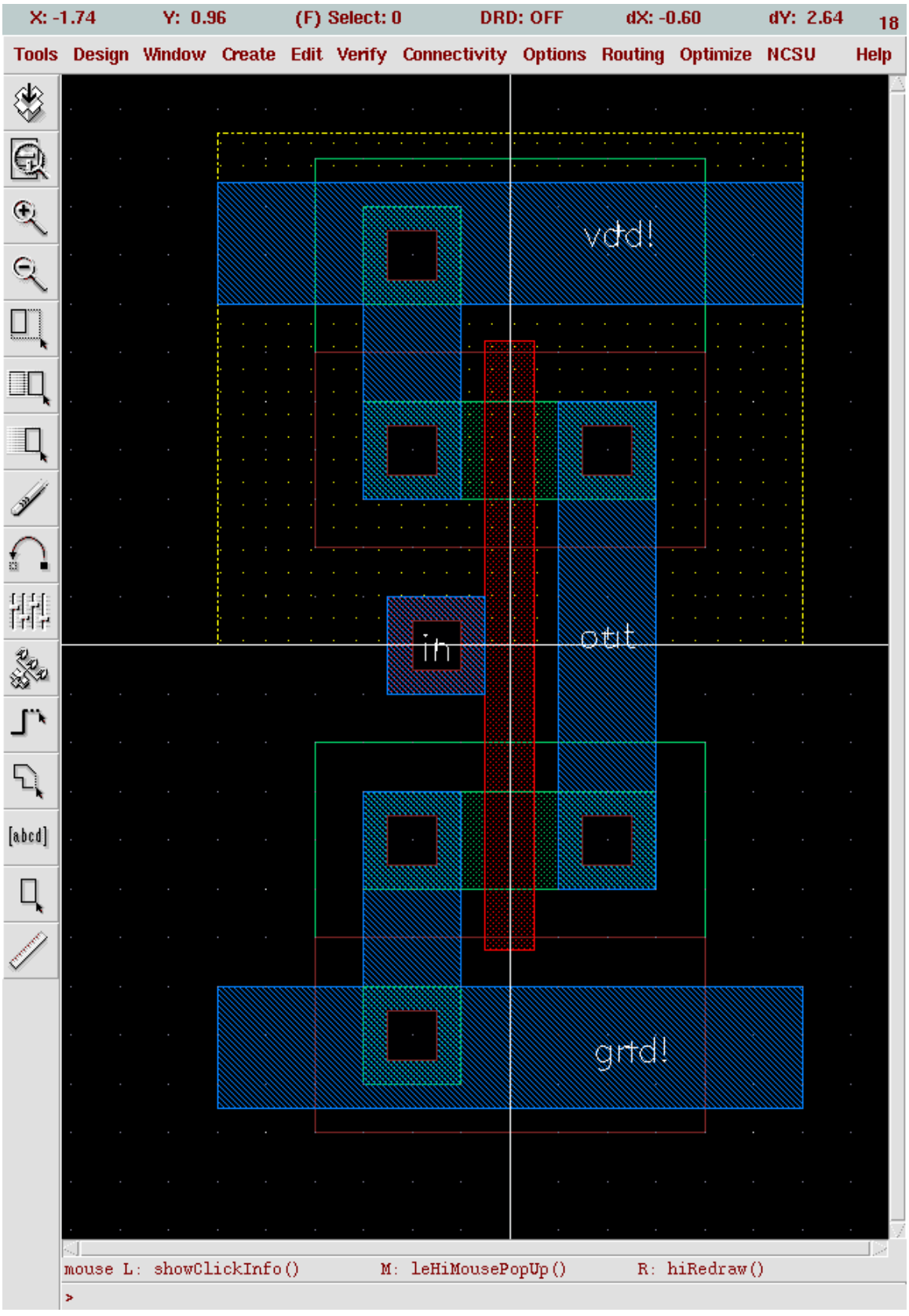
Layout

Now that you have a schematic view for the inverter and verified that it functions as you want we can begin creating the layout view. To do this go back to the Library Manager and select the inverter library and the inverter cell. Then select *File->New->Cellview.* From the pull down menu chose Virtuoso. This should open two windows. The first is the Layer Selection Window (LSW) and the second is the actual layout view.

Now, look now at the LSW (Layer selection window). This window shows you the names of the layers that are “valid” (meaning that you can manipulate them). You can make different layers visible and in-visible. To toggle a layer’s visibility, middle-click on the name of the layer in the LSW. You can make all layers visible with the “AV” button, and no layers visible with the “NV” button. You may have to force the window to redraw with *Window->Redraw* command (ctrl-r) for the changes to have effect.

Note that even if you make all layers invisible, you may still see some shapes. This is because not all layers are “valid”. Shapes in invalid layers cannot be altered and are always visible. To make all layers valid, you can choose Edit->Set Valid Layers… in the LSW. In general, it is recommended that you not set all layers as valid, because this clutters up the LSW with many unused layers.

Towards the top of the layout window, you should see X and Y coordinates of the cursor in the layout window being continuously tracked. You’ll notice that minor grid points are 2.0 lambda (0.24um) apart and that the cursor snaps to every 0.5 lambda 0.06um. This will make for easier editing for the lambda-based rules you will be using. From the design rules and feature sizes, we can estimate the minimum size of the inverter to be about 24 lambda wide by 41 lambda tall (counting the nwell and selects). An example inverter layout is shown below. Note that all contacts (small black squares) are layer cc.


X: -1.74
Y: 0.96
(F) Select: 0
DRD: OFF
dX: -0.60
dY: 2.64
18
Tools
Design
Window
Create
Edit
Verify
Connectivity
Options
Routing
Optimize
NCSU
Help
vdd!
in
out
grtd!
mouse L: showClickInfo()
M: leHiMousePopUp()
R: hiRedraw()
>

You are now ready to start adding/editing shapes in the layout window. Let's begin by looking at the common editing commands and their shortcut key strokes. Select the *Create* menu. We can now see that we can add Polygons, Rectangles, and Paths. Lets experiment with each option. First lets draw a simple rectangle. Go to the LSW and select the layer you wish to draw. Then select *Create->Rectangle* or simply press the "r" key. Now click where the first corner is, then click where the opposite corner is. The shape should have been drawn. You can also draw a polygon, by clicking out each vertex of the shape. Or you could create a path where you specify the width in the property box. Practice drawing some shapes now.

Now that you have learned to draw shapes, it is important to be able to edit them. The most basic operations you will need are the copy, move, stretch, and reshape commands. They can be found under the edit menu. There are many advanced tricks you can use for copying a set of objects multiple times at constant x and y offsets, or to move a objects a specified amount. Most of these are available in the form that appears for each command. As the semester progresses you will become more familiar with some of these features and you might want to play around with them more.

There are many tricks you will learn for using the layout editor, but to describe them all here would take to long and distract you. We suggest that you take some time as you use the tools to explore the features available as you learn more. Use some time now to get familiar with the placement and editing of the blocks.

Pitch Matching and Pins

Usually cells (like the inverter) are designed so that they can be assembled side by side such that the power and ground lines abut each other. This is called *pitch matching*. This reduces need for some amount of routing and also enables a more orderly and regular layout. The pitch is therefore a standard value which you have to decide on at the onset. Let us assume that this height is 38λ (a truly tiny pitch that is really probably too small for normal usage). This is the height measured from the top of the power rail to the bottom of the ground rail.

Draw a 5λx24λ Metal1 object to represent the ground rail in the inverter design. You can now do this by deleting all the objects you have been drawing and creating the power rail. Note that gnd and vdd are special nets (along with the input and output nets). LVS and parasitic extraction and simulation require that we identify these nets. This is done using pins and text. We could just draw all of the shapes and place text and pins later via a number of different methods. The one that I think gives the cleanest look uses the *Create->Pin* command. We will use this to create our gnd rail instead of *Create->Rectangle.* Note that the net names for gnd and vdd are gnd! And vdd! (with an exclamation mark) but for normal io nets there is no !)

Select metal1. Choose *Create->Pin* and choose:

- Name : gnd!
- Mode : rectangle (you may have to choose Mode: shape pin to have this option appear)
- Display Pin Name : checked
- click "Display Pin Name Option…" and choose height 0.12 (1 lambda height for the text)
- I/O Type : inputOutput (input and output signals will use the appropriate setting)

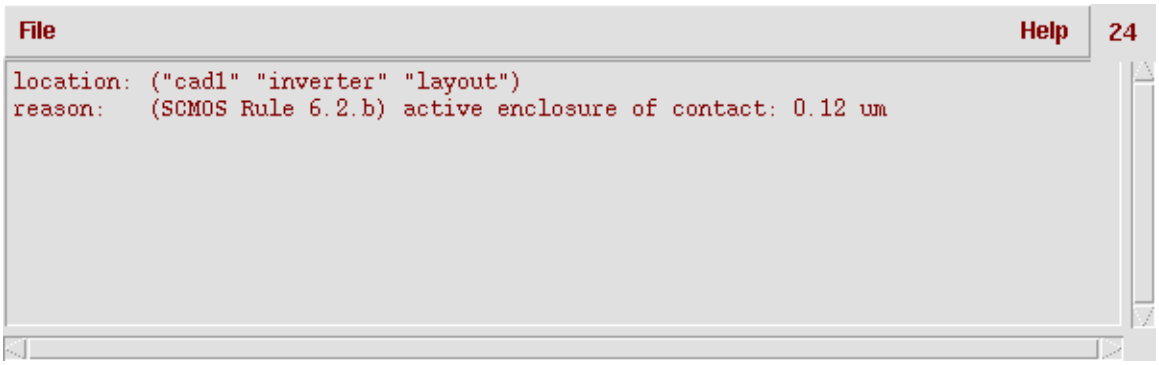
Now place two opposite corners of the rectangle like normal and then place the label somewhere inside. This shape can be used just like any other shape except that the label will move with it and if you edit its properties the connectivity tab will be enabled and will allow you to edit the terminal name and I/O type.

Using the coordinate tracker you can now place your vdd line. Similarly place all other shapes as illustrated in the layout and remember to follow the design rules. Use the editing features described above if you make mistakes or need to copy or move objects.

Design Rule Checking (DRC)

To perform a Design Rule Check (DRC), choose *Verify->DRC.* The DRC form appears. Make sure that the "Echo Commands" box is unchecked, or DRC will take a lot longer to complete (because it will write lots of text to the CIW). For larger designs than this one, this text can be a nice "status indicator" to show how far the DRC check has progressed, but for now, it just makes things slower. Click OK to run DRC.

If you look in the CIW it should list the number and types of violations that you receive. It will also create Markers in your layout view to show you the errors. You can learn about the errors by selecting *Verify->Markers->Explain* and clicking on a marker. Alternatively, you can select *Verify->Markers->Find…* and click "Next" to cycle through all of the error. Errors are described in a marker text window like this one.



In this particular case, a contact was too close to the edge of the active. You will know that there are no errors when there are no markers and you see the following message in the CIW:

********* Summary of rule violations for cell "inverter layout" *********
Total errors found: 0

To learn more about each design-rule, you can look it up in the design rules provided on the website. Note the rule number will correlate to the one in the handout, but the measurements in the marker text are just lamda*.12um. Remember that you can measure distances by drawing a ruler (you can hit "k" to draw a ruler). You can delete all of the rulers that you have drawn by hitting SHIFT-K. If you simply want to remove the error markers, choose *Verify->Markers->Delete All…*

The only errors we allow in the DRC check is for nets with the same label not being connected, and only for VDD and GND. This is because we will put off connecting all the VDD and GND rails until we get to the top level of our design. If we figure out how to eliminate this we will let you know.

Layout vs. Schematic (LVS)

When you are done with DRC for your layout, save the design, and then select *Verify->Extract*. Make sure Extract Method is set to "flat", Echo Commands is unchecked, and Join Nets with Same Name is checked. Click Ok. You should see something like the following messages appear in the CIW:

```
Extraction started.......Mon Sep 13 19:33:07 2004
           completed ....Mon Sep 13 19:33:08 2004
   CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
******** Summary of rule violations for cell "nand2 layout" *********
  Total errors found: 0
```

You should also notice that a new cell view has been created, called “extracted”. Open the extracted view and examine it. You’ll notice that the layers in the extracted view are like the ones in the layout view (metal1, poly, etc.) except that they’re in the “net” purpose (“nt”) rather than the “drawing” purpose (“dg”). Cadence uses the different “purposes” to distinguish shapes that are meant for different things. Here, for example, the net purpose is used to indicate that the shape is showing the location for a net, rather than being the actual shape that should be used to generate the mask-patterns for manufacturing.

Select the metal1 shape at the top of the layout and look at its properties. Click the “Connectivity” button in the properties dialog box. You’ll notice that the Net Name is set to “vdd!”. This is showing that the extractor recognized the labels in the layout and assigned net names according to the labels. Check the other shapes in the design, and you’ll find that all of the label names that you used are preserved.

We need to set up some parameters for the LVS. Do this by selecting *NCSU->Modify LVS Rules.* In the window that appears make sure the “Compare FET parameters” box is selected so that the LVS check makes sure that you have sized the transistors the same in layout as you did in your schematic.

Now you are ready to run LVS, do so now by selecting *Verify->LVS*

NOTE: If you are running LVS multiple times, you may see a dialog box warning you that the “LVS Run directory does not match the Run Form”. If you select the “Form Contents” option, then the LVS dialog will be updated with the name of the cell that you are currently editing. Otherwise, if you choose the “Run Directory” option, the dialog will be updated with the values from the last LVS run.

Set the LVS form to point to your schematic and extracted view. Then click the “Run” button. In the CIW, you will see the message “LVS job is now started...” After a while, you should see a dialog box appear that says the job is complete.

This does NOT mean that the LVS check was passed… only that the LVS check completed. To find out if LVS was successful, click the “Output” button in the LVS form. You should not assume that LVS was successful unless you see the message **“The netlists match”** in this output. Just as an example of what can go wrong when running LVS, in case yours worked on the first try, remove the piece of metal connecting the PMOS transistor to VDD and extract and run lvs again.

You can use the Error Display (as long as you started LVS from within the extracted view) to see the errors. The functionality of the Error Display is similar to the Markers for DRC. However, LVS discrepancies are often more difficult to debug than DRC errors. Note also that the layout text is important for the check. Mislabeled nets will lead to port discrepancies and perhaps even to device and net mismatches.

Debugging LVS errors on the inverter is simple since there are so few nodes and devices to deal with. Things can get a lot more complicated even when you’re dealing with only 10-20 devices and 8 or 10 nodes. A simple mistake like shorting vdd to gnd can lead to very few devices matching up and the information in the report file may not immediately lead you to the problem area. It helps to understand a bit about how LVS goes about comparing the layout and the schematic when debugging more complex circuits. First, the ports and any named internal nets are taken as initial correspondence points. The program will work its way in from these initial correspondence points, attempting to match nodes by the number and types of devices and pins that connected to them. Similarly, devices are matched if their pins share similar node connections. When LVS is unable to make a complete match, it will report nodes and devices that don’t match up. The report will show the source (schematic) device or node in one column and the layout device or node it attempted to match it to in another column.

A good place to start debugging is to look at the area of the report file called Netlist Summary. See if you’re at least getting the same number of nmos and pmos devices between layout and schematic. Check the node counts too. Shorts in the layout will show up as fewer nodes in the layout while opens in the layout will show up as extra layout nodes. Again, sometimes simple mistakes (vdd and gnd shorts)

can lead to many mismatches. You may have hundreds of discrepancies that will eventually disappear after making 3 or 4 layout changes. Take advantage of the initial correspondence points for tough debug problems.

You may also have parameter mismatches, that is the width or length of your transistors don't match.

Once you have a successful LVS you can save the report by selecting *File->Save* in the output display window.A successful LVS should contain something similar to the following (with a long command line at the top and two lists of files after):

```
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.

    Net-list summary for /l/cad1/LVS/layout/netlist
       count
        4               nets
        4               terminals
        1               pmos
        1               nmos

    Net-list summary for /l/cad1/LVS/schematic/netlist
       count
        4               nets
        4               terminals
        1               pmos
        1               nmos

The net-lists match.

                            layout  schematic
                               instances
        un-matched             0       0
        rewired                0       0
        size errors            0       0
        pruned                 0       0
        active                 2       2
        total                  2       2

                                 nets
        un-matched             0       0
        merged                 0       0
        pruned                 0       0
        active                 4       4
        total                  4       4

                               terminals
        un-matched             0       0
        matched but
        different type         0       0
        total                  4       4
```

Layout Parameter Extraction

Now we're going to extract the wire capacitances from the layout. Select Verify->Extract… . Set the options as before, but this time click the "Set Switches" button. In the dialog box that appears, select "Extract_parasitic_caps" and click Ok. You should see that string appear in the "Switch Names" field of the Extract form. Click Ok. Again, you should see the confirmation in the CIW that the extraction completed successfully. You should also see some lines indicating that wire capacitors were created, such as the following:

4 pcapacitor ivpcell NCSU_Analog_Parts parasitics created.

To annotate the parasitics into the extracted view you need to go to the LVS menu. *Verify->LVS*. Inside make sure the correct extracted and schematic view are listed, then click the backannotate button and a new dialog should appear. First click the Add Parasitics button to annotate the views. Next we want to view the parasitics so click the print all… button and click OK in the dialog that appears. It should open a window that displays the parasitics that were calculated. For this design you should only see parasitic capacitances in the list.

Simulation of Extracted Netlist

Now that we have extracted the parasitics and back annotated them into our schematic we will want to run the Spice simulation once again to get the final delay and rise/fall times. To do this open the schematic view and start the Analog Design Environment. Once you are inside click *Setup->Environment.* In the switch view list add extracted just before schematic if it is not already there. Now you can run the HSpice simulation as before to measure the values.

Digital Delay Annotation and Resimulation

At this time we are not certain what the best way to do this is. We hope to have a solution in place before CAD2.

Conclusion

This is the basic flow you'll use for full-custom design in your EECS 427 project. Keep this tutorial on hand while you're doing your CAD assignments for your projects since it contains a lot of useful information. Also feel free to explore better ways of doing things in the tools. There are many shortcuts and alternative ways of doing things in the tools and we're unable to cover them all in this document.

Appendix A: Cell Naming Conventions

When naming cells (e.g. inverter), nets (e.g. in, out) and instances use the following naming conventions:
1) Names can contain only lower case alphabet, digits or the underscore: "_".
2) Names should begin only with lower case alphabet.
3) Names should not end with "_".
4) Don't use names of verilog keywords.
5) Don't use names that are for VTVT standard cells, the common ones are given in the list below.
By following these naming conventions you can avoid problems in other tools that use the design data you enter in Design Architect. In addition, the following list of names cannot be used as cells you create. They are either cells that exist in our standard cell library, in our memory components that will be used later in the semester, or are verilog keywords. Also, do not name two different cells with the same name even in different cad assignments. You will later be combining all your cad assignments into one project so this will cause problems also. If you follow these rules, you'll save a lot of problems from occurring later in the semester.

Illegal Cell Names (VTVT Standard Library Cell Names)

buf_[1,2,4] (Shorthand for buf_1, buf_2, and buf_4)
inv_[1,2,4]
and2_[1,2,4]
and3_[1,2,4]
and4_[1,2,4]
or2_[1,2,4]
or3_[1,2,4]
or4_[1,2,4]
nand2_[1,2,4]
nand3_[1,2,4]
nand4_[1,2,4]
nor2_[1,2,4]
nor3_[1,2,4]
nor4_[1,2,4]
xor2_[1,2]
xnor2_[1,2]
mux2_[1,2,4]
mux3_2
mux4_2
ABnorC
ABorC
ab_or_c_or_d
not_ab_or_c_or_d
dec2_4
dec3_8
fulladder
bufzp_2
invzp_[1,2,4]
cd_8
cd_12
cd_16
lp_[1,2]
lrp_[1, 2, 4]
lrsp_[1, 2, 4]
dp_[1,2,4]
drp_[1,2,4]
drsp_[1,2,4]
dksp_1
dtsp_1
dtrsp_2
jkrp_2
filler

Appendix B: Hierarchical Layout Design

For the inverter, nothing in this paragraph needs to be done, but refer back to it for CADs. On most of the CAD assignments, you will need to include cells that you have previously designed. Virtuoso allows you to do this with the *Create*->Instance menu option. When you select this you will be able to place an instance of another layout. You are also able to select the orientation (flip, rotate, mirror) by either editing the properties ("q" key) or clicking the buttons in the instance creation dialog box. When you place the instance you will not be able to see what is inside of it. This is because you are not able to edit the cell from this layout, but you can peek inside of it by enabling how deep you want to see into the hierarchy. Or you could display all levels by pressing "Shift-f". To turn off all levels press "Ctrl-f". If you only wish to look at a few layers of depth, you can do this by going to *Options*->Display and playing with the Display Levels. Similarly you can place instances of your schematic symbols in other schematics to create hierarchical designs.

Appendix C: Connecting from home

First we will deal with getting the right software then how to connect effectively.

Linux : Nothing extra needed

MAC : Install X11 : http://www.apple.com/downloads/macosx/apple/x11formacosx.html

Windows : Install Putty http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html (other ssh clients work fine too) and XMing http://sourceforge.net/projects/xming (other X servers work fine too)

Now find a machine to log into. The safest bet is login.engin.umich.edu but that may be overloaded. If you ssh to login.engin.umich.edu (via:
Linux : % ssh <uniqname>@login.engin.umich.edu

MAC : % ssh <uniqname>@login.engin.umich.edu (from within an xterm from X11 or terminal)

Windows : Use putty

On login.engin.umich.edu type:

```
% hostinfo –linux
```

This will provide you with a list of CAEN linux lab machines with low load. However, if you use one be warned that they can be rebooted by users in the lab at any time so save often.

Now that you have picked a machine (a lab machine or the login servers) connect to it (using the full hostname):

Linux : % ssh –Y <uniqname>@<hostname>

MAC : Start X11 and start an xterm. Enter the command from within the xterm. It will not work from terminal
%ssh –Y <uniqname>@<hostname>

Windows : start XMing (the default rootless setting is fine)
Start putty. Enter the hostname and your uniqname but before you click open check the following check box which can be browsed to using the pull-downs on the left of the putty window:
Connection->SSH->Tunnels->Enable X11 Forwarding

In all cases you should get a terminal window connected to the target machine from which you can start more xterms via :

```
% xterm &
```

Or start all of the GUI and command line tools from as normal.