The Design and Simulation of an Inverter

(Last updated: Sep. 1, 2010)

A. 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 *Composer*. **(Section C)**

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 is also done using *Composer*. **(Section D)**

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. **(Section E)**

4. Create an analog testing schematic, also with Cadence *Composer*. **(Section F)**

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. **(Section F)**

6. Physically ***layout*** the inverter according to some CMOS process rules. In our case we will be using the IBM 0.13 micron CMOS process with MOSIS SCMOS DEEP SUBM design rules available as a separate handout. Layout is done using the Cadence *Virtuoso Layout Editor*. **(Section G)**

7. Check the layout to verify that it conforms to the process design rules. This is done using Mentor Graphics *Calibre* ***DRC***, which is accessible from within Cadence Virtuoso**. (Section H)**

8. Check that the layout view and schematic view match using Mentor Graphics *Calibre* ***LVS***, a layout vs. schematic checker that is also available from Virtuoso. **(Section I)**

9. Extract parasitic layout parameters such as capacitances which contribute to circuit delays. This is also done using *Calibre* ***xRC*** from within *Virtuoso*. **(Section J)**

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

11. Annotate the schematic to reflect the circuit delays you calculate in ***HSpice***. **(Section L)**

12. Simulate with ***NC Verilog*** again with the new digital circuit delays. **(Section L)**

B. Before you start

Environment 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 are in appendix D 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 unique name 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 be given for the command line or for the specific GUI applications. 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, gvim, vi, emacs, etc.)

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

Add the following lines to the bottom of the file:

```
if ($?prompt) then
  if (! -f /afs/umich.edu/class/eecs427/ibm13/setup/cshrc) then
   gettokens
  endif
  source /afs/umich.edu/class/eecs427/ibm13/setup/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/f10/<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
```

Create your working folder

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
% mkcdkdir cad1
% cd cad1
```

The above command is equivalent to the following command:

```
% cd ~/eecs427
% mkdir cad1
% cp $CDK_DIR/cds.lib $CDK_DIR/display.drf $CDK_DIR/calview.cellmap cad1
% ln –sf $CDK_DIR/.cdsinit cad1/.cdsinit
% ln –sf $CDK_DIR/.simrc cad1/.simrc
% mkdir cad1/Calibre
% mkdir cad1/Calibre/DRC
% mkdir cad1/Calibre/LVS
% mkdir cad1/Calibre/xRC
% mkdir cad1/Calibre/runset
% cp $CDK_DIR/*.runset cad1/Calibre/runset/
% cd cad1
```

Starting Cadence

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*. If the Library Manager window does not appear, you can select ***Tools->Library Manager…*** in the CIW window to open it. The Library Manager allows you to browse the available libraries and create your own.

C. Create the Inverter Schematic

In the CIW, create new library called inverter. Select ***IBM_PDK->Library->Create***. This will open new dialog window, in which you need to enter the name of your library and library location. Enter "CDSLIB" for the name (don't change the location), and for the "Technology File" section, choose "Attach to existing techfile", then press OK. A new window with the label "Attach Design Library to Technology File" will pop up, and you should set the Technology Library to "cmrf8sf", and then click OK. You will be prompted with a new window named "Add AMS Library Properties", and you should set the "Number of levels of metal" to "6-2" via the pulldown menu and click OK. Do not create a new library using the Library Manager.

You should now see the library "CDSLIB" appear in the "Library" section of the Library Manager window.

Next, select the library you just created ("CDSLIB") 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 Name. 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.

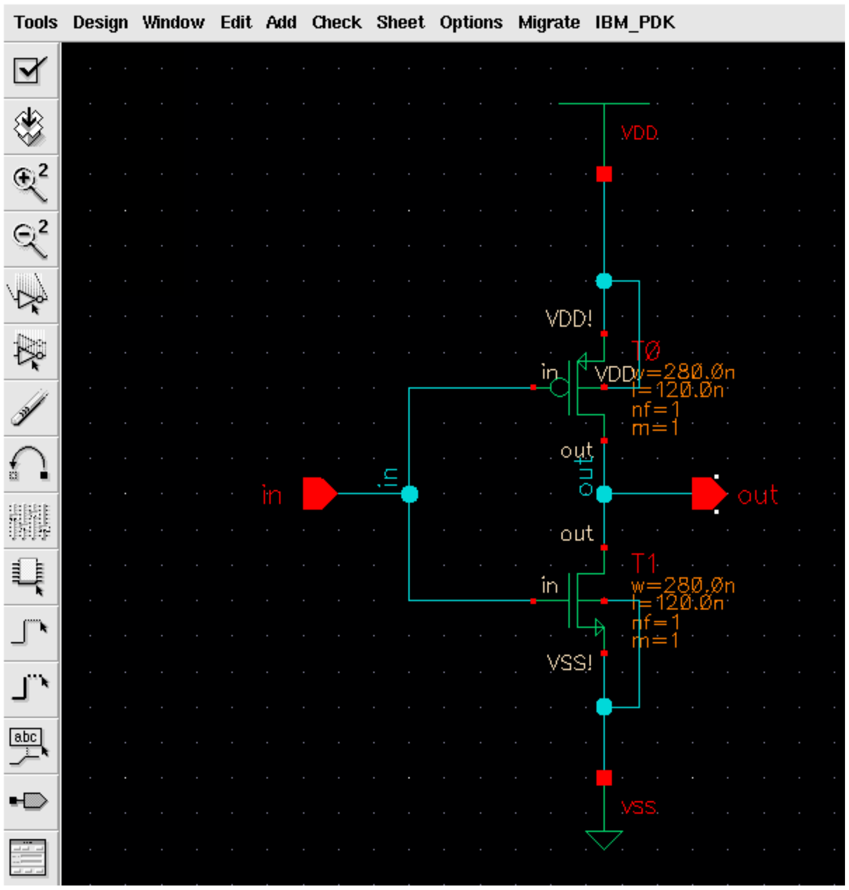


Fig. Inverter schematic

(NOTE: Pay attention to the shortcuts listed in this document. In many cases, they are also listed next to the command in the submenu. Learning the shortcuts can improve your schematic/layout efficiency.)

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

From the Schematic Window, choose ***Add->Instance*** (Shortcut: press the 'i' key). The respective cell name for the components that we will be using is shown below:

	Library	Cell
N Transistor:	cmrf8sf	nfet
P Transistor:	cmrf8sf	pfet
Supply Nets:	eecs427Lib	VDD, VSS

Place instances – In the "Add Instance" window, select the library and then click on the cell name for the component that you want to place. You will then see your component in the Schematic Window in yellow; you can move your mouse to move the component to a desirable location, then click the left button to place the instance. You can press escape to close the "Add Instance" window once you finish placing instances. You can perform basic operations such as "copy" or "delete" to the instances you just placed by selecting the instances and then clicking ***Edit->(operation)***.

Place IN pin – From the Schematic Window menu, select ***Add->Pin*...** (Shortcut: 'p') 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)*** (Shortcut: 'w'). 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*** (Shortcut: 'X' - note the capitalized case).

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

Schematic check completed with no errors.
"CDSLIB 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 the width of your transistors will probably be **160n**, and the lengths will be **120n**. 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 **280n** and length **120n**. And check and save your design again.

D. 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.

E. 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 testfixture.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 definition 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 an apostrophe (’), 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 referenced 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 testfixture.verilog file). 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 VSS. 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 (T0 and T1), and see they have drain, gate, source, and body ports (D, G, S, and B, respectively).

Now that you have learned to explore the design in the browser, let’s go back to the test level by clicking on test. Let’s 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 Window”.

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 from CDSDOC. To invoke “cdsdoc” run the following commands:

```
% ncverilog term &
(new terminal window will open; in new window, execute the next command)
% cdsdoc
```

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 few 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 from *Command* to *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 *NC Verilog* and *SimVision* windows for now.

F. 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 CDSLIB 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.

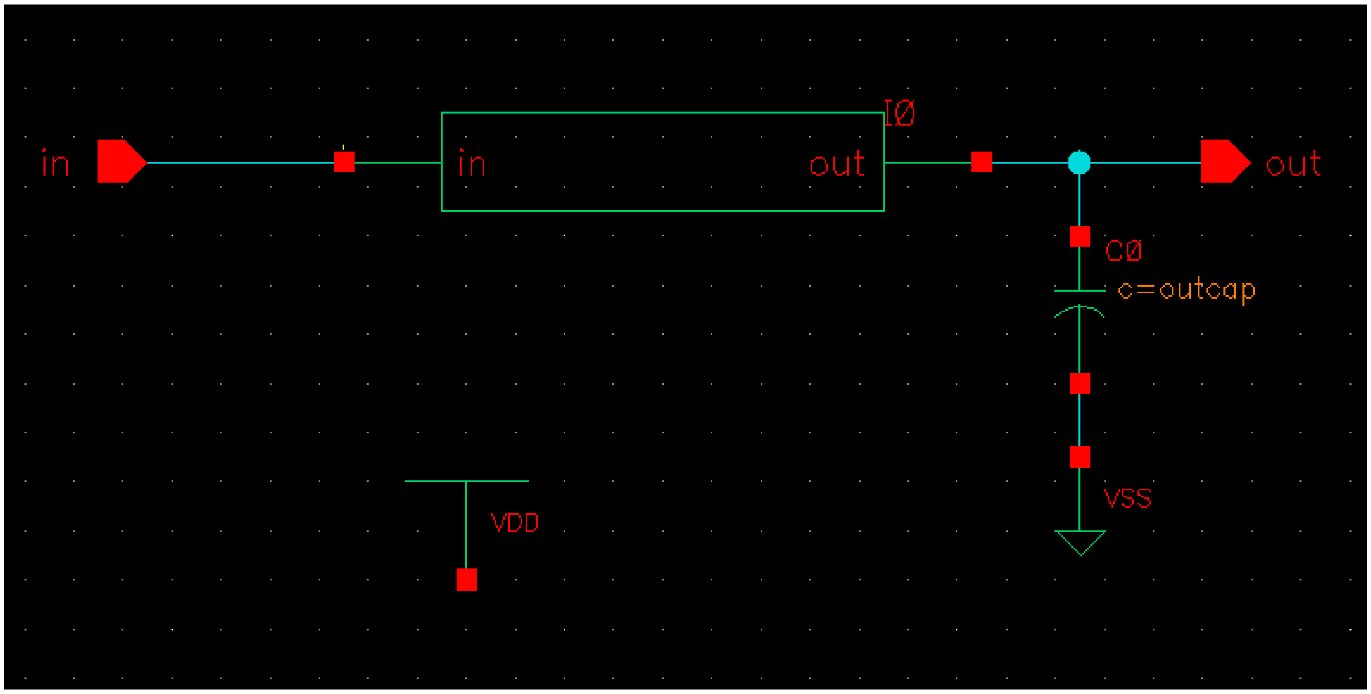
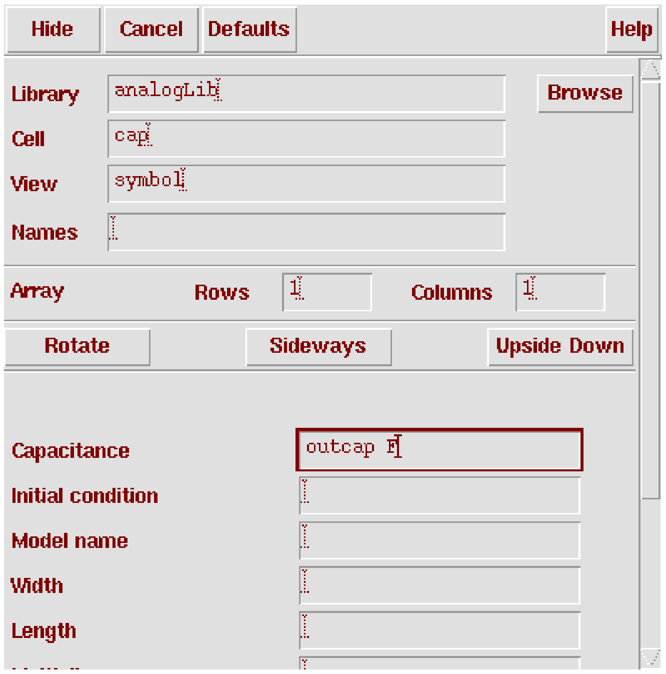


Fig. Inverter test schematic

Place inverter – Add your inverter symbol the same way as you add NMOS and PMOS in schematic. The inverter symbol is from your CDSLIB library. Place VDD, VSS, in pin, out pin, and wires the same way as explained before.

Place output cap – Add a capacitor in between your output pin and VSS. Capacitor can be found under the library “analogLib” and cell name “cap”. Remember to change the Capacitance value “outcap F” as shown below. This will allow you to change your output cap value freely during simulation. Remember 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.



Library	analogLib
Cell	cap
View	symbol
Names	
Rows	1
Columns	1
Capacitance	outcap F
Initial condition	
Model name	
Width	
Length	

Fig. Adding capacitor

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 Design Environment Window.

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...*** Click on "tran" for Analysis, and then fill out the form with the following values:

Start: 0 Stop: 10n Step: 100p

Next we will add the variable from our schematic (outcap). From the Analog Design Environment menu, select ***Variables->Copy From Cellview***, note that "outcap" will appear in the "Design Variables" panel. Then select ***Variables->Edit***; the Editing Design Variables form will appear. Click on "outcap", and then fill out 25f as the value, and then click OK.

Now select ***Outputs->To Be Plotted->Select On Schematic*** to choose the nets that we want to plot. This will bring up the Schematic window again. First select the "in" net, then select the "out" net (note that both nets will be highlighted). Now go back to the Analog Design Environment window, you should see both "in" and "out" are shown in the "Outputs" panel.

Now we need to set up the stimuli. Select ***Setup->Stimuli…***; the "Setup Analog Stimuli" window will appear. First select "Inputs" for Stimulus Type, then click on "Enabled" and change the Function to "pulse". Now fill out the forms as follows:

Voltage1: 0.0 Voltage2: 1.2 Delay time: 1n Rise time: 100p Fall time: 100p Pulse width: 2n Period: 4n

Then click on "Change". Next, click on "Global Sources" for Stimulus Type, and then choose VDD, click on "Enabled" and change the DC voltage to 1.2. Choose VSS, click on "Enabled" and change the DC voltage to 0. Click on "Change" again to save all the setup. Now click on OK to exit the Stimuli setup.

To avoid the trouble of having to key in all the values again next time, you can save the setting by selecting ***Session->Save State***. The "Saving State" window will appear. First click "Cellview" for the Save State Option, then click OK. This will save all the setting as part of the CDSLIB library. So when you open Analog Design

Environment next time, just simply select ***Session->Load State***, then change the State Option to “Cellview”, then you can recall all the setting from last time.

Now we are ready to run the simulation. From the Analog Design Environment 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.

A pulse plot should appear showing you the input and output pulse. 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 the delta cursors on the 50% (0.6v) of both the input and output waves. You do this by selecting ***Trace->Delta Cursor***. Two markers will be placed; you should move one of the marker to 50% of input and one marker to 50% of output. The dx and dy value will be shown below, and the absolute value of dx will be your 50% gate delay. 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.

G. Layout

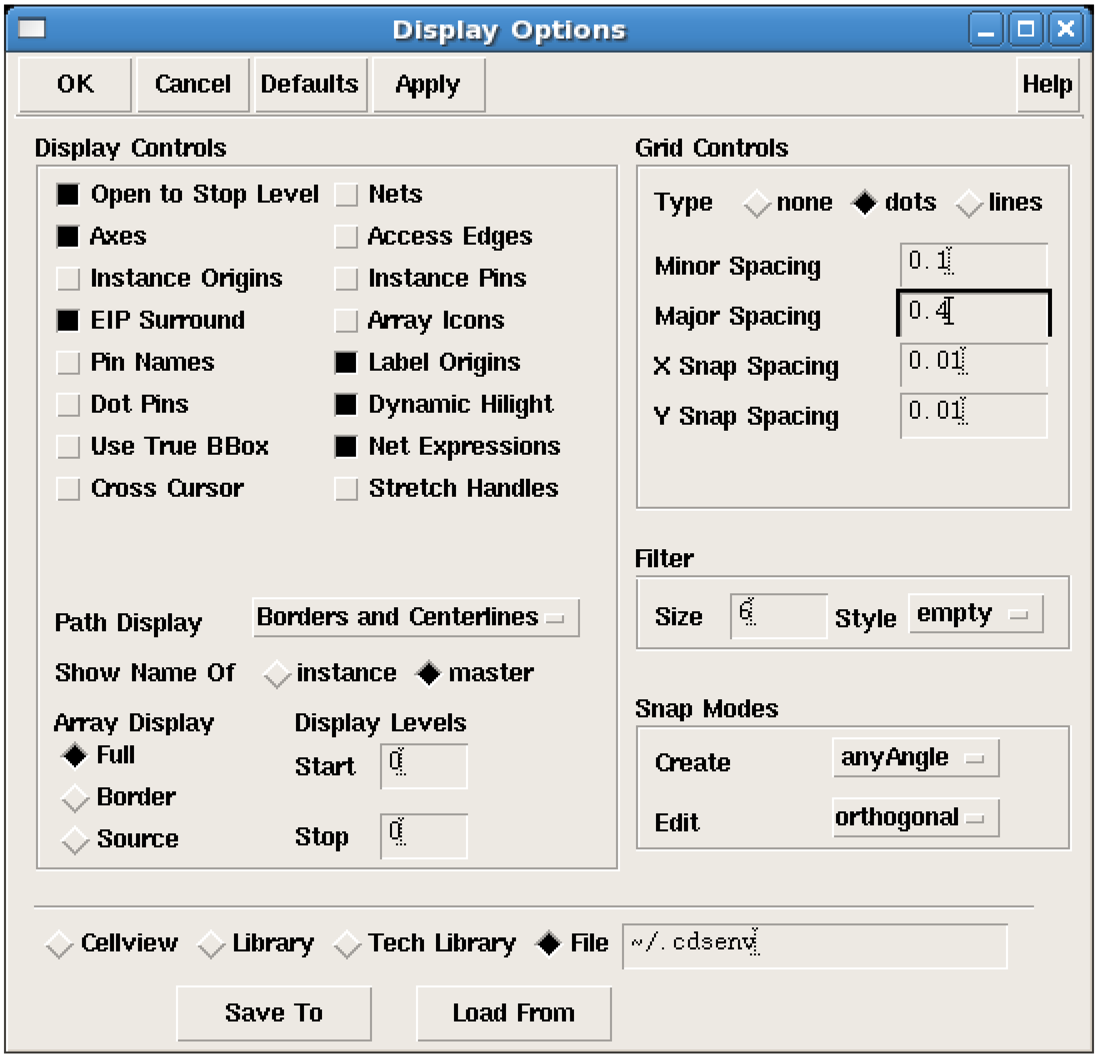
Now that you have a schematic view for the inverter and verified that it functions as you want we can begin creating the layout view. (Please refer to **Appendix A** for the cross view of an inverter before drawing the layout.) To start the layout, go back to the Library Manager and select the inverter 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 (Shortcut: "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. A list of layer definitions is shown below:

Layer	Color	Purpose	Discription
RX	Green	drw	Active area (gate oxide and N+/P+ diffusion regions)
PC	Red	drw	Polysilicon line
NW	Yellow	drw	N-well
BP	Brown	drw	P+ select (area which are blocked from n+ source/drain implant; defines PMOS and p+ substrate contacts)
CA	Yellow	drw	Metal contact (connects PC or RX to M1)
Mx		drw	Metal layer (x = 1, 2, 3, 4, 5,or 6)
Vx		drw	Via (x = 1, 2, 3, 4, or 5)
GRLOGIC	Pink	drw	Shape identifying structures that are exempt from more stringent design rule checks
Mx		lbl	Metal label layer (x = 1, 2, 3, 4, 5,or 6) used to label input, output, VDD, and VSS pins
OUTLINE	White	drw	Define the cell boundary (optional)

Towards the top of the layout window, you should see X and Y coordinate of the cursor in the layout window being continuously tracked. The unit of the coordinate is micron (μm). You can change the grid spacing and grid snap according to your preference; a recommended spacing value is to use 0.4μm for major grid, 0.1 μm for minor grid, and 0.01 m for grid snap. You can change the spacing value by pressing 'e' or click on ***Option->Display…***. The Display Options window will appear, as shown on the next page; please note that once you change the spacing value, you should save the changes by clicking on "File" on the bottom of the Display Options window, and then click on "Save To". Doing so will save you from the trouble of changing the grid value every time you open Virtuoso. Another thing to note here is that "Display Levels" will help you control how many hierarchies would be displayed in the current layout window. For example, if you have display levels start at 0 and stop at 0, only the top level detail would be displayed in the Layout view. If you have display levels start at 0 and stop at 20, layout details from all hierarchies would be displayed. You won't need this feature in this tutorial but you will use it in cad1.



Display Controls

■ Open to Stop Level	☐ Nets
■ Axes	☐ Access Edges
☐ Instance Origins	☐ Instance Pins
■ EIP Surround	☐ Array Icons
☐ Pin Names	■ Label Origins
☐ Dot Pins	■ Dynamic Hilight
☐ Use True BBox	■ Net Expressions
☐ Cross Cursor	☐ Stretch Handles

Path Display: Borders and Centerlines

Show Name Of: instance [✓ master]

Array Display: [✓ Full] Border Source

Display Levels: Start 0, Stop 0

Grid Controls

Type: none [✓ dots] lines

Minor Spacing	0.1
Major Spacing	0.4
X Snap Spacing	0.01
Y Snap Spacing	0.01

Filter: Size 6, Style empty

Snap Modes: Create anyAngle, Edit orthogonal

Cellview Library Tech Library [✓ File] ~/.cdsenv

Layout display options window

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 keys. Select the ***Create*** menu. We can now see that we can add Polygons, Rectangles, and Paths. Let's experiment with each option. First let's draw a simple rectangle. Go to the LSW and select the layer you wish to draw (**RX | drw**, for instance). 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 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 too 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.

Your inverter layout should look similar to the picture shown below. Please note that VDD and VSS net should be approximately 3X wider than minimum metal1 width.

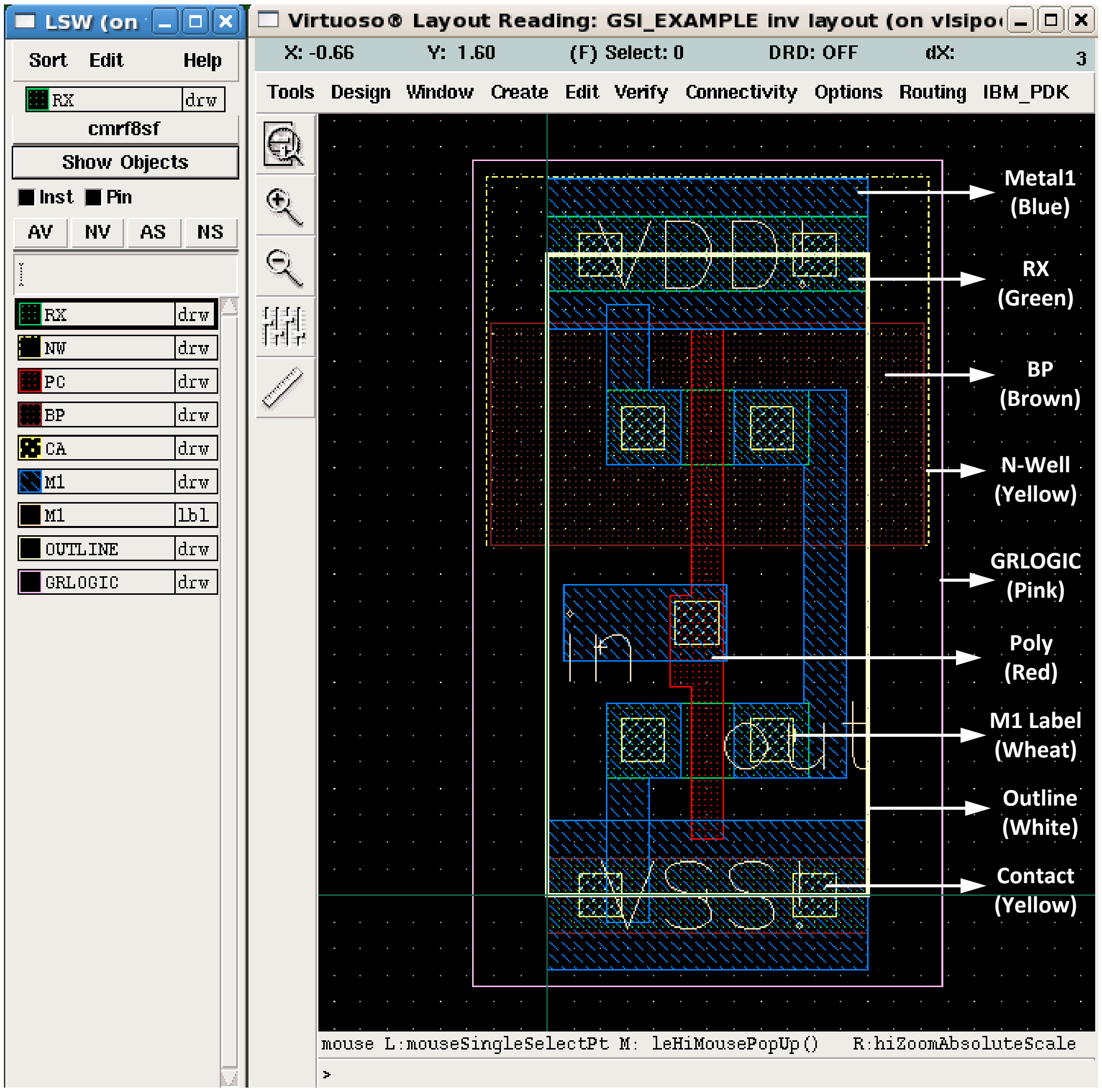


Fig. Inverter layout

You should follow the Layout Rules to construct all of your layouts. Please note that this is a simplified version of the design rule. For the more complete (and more complicated) design rule, please refer to chapter 3 of the IBM design manual, which can be found in ***ibm13/cmrf8sf/V1.3.1.5LM/doc/cmrf8sf.design_manual.pdf*** under the class directory.

A good place to start the layout is the nfet and pfet, since you know the dimension of the length and width of the devices. Then construct the contact next to the gate following the design rules, and connect the device with metals. Finally, define your n-well and BP when you are all done. Use the design rules in the following figure to get a sense of the layout size.

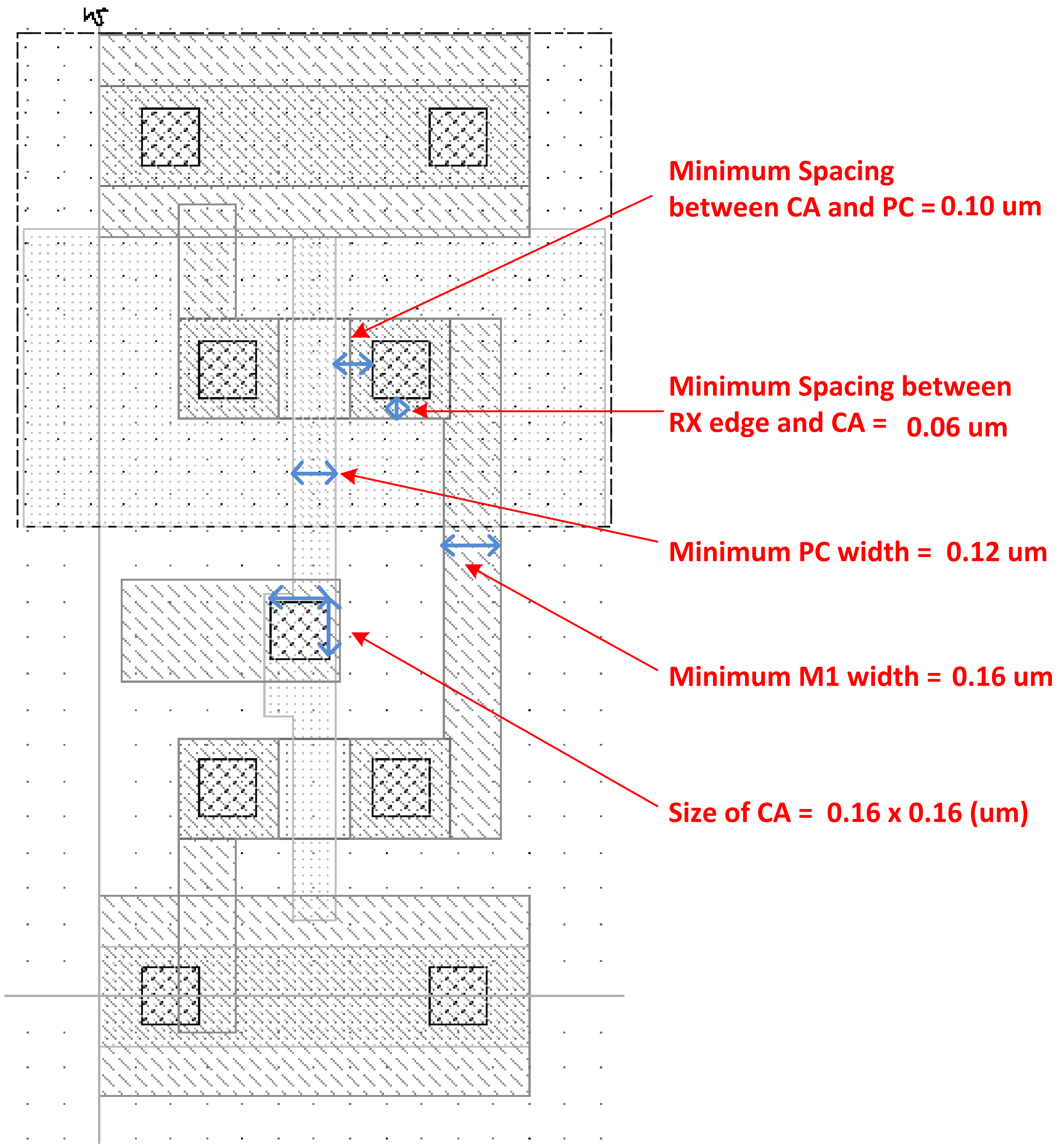

Minimum Spacing between CA and PC = 0.10 um
Minimum Spacing between RX edge and CA = 0.06 um
Minimum PC width = 0.12 um
Minimum M1 width = 0.16 um
Size of CA = 0.16 x 0.16 (um)

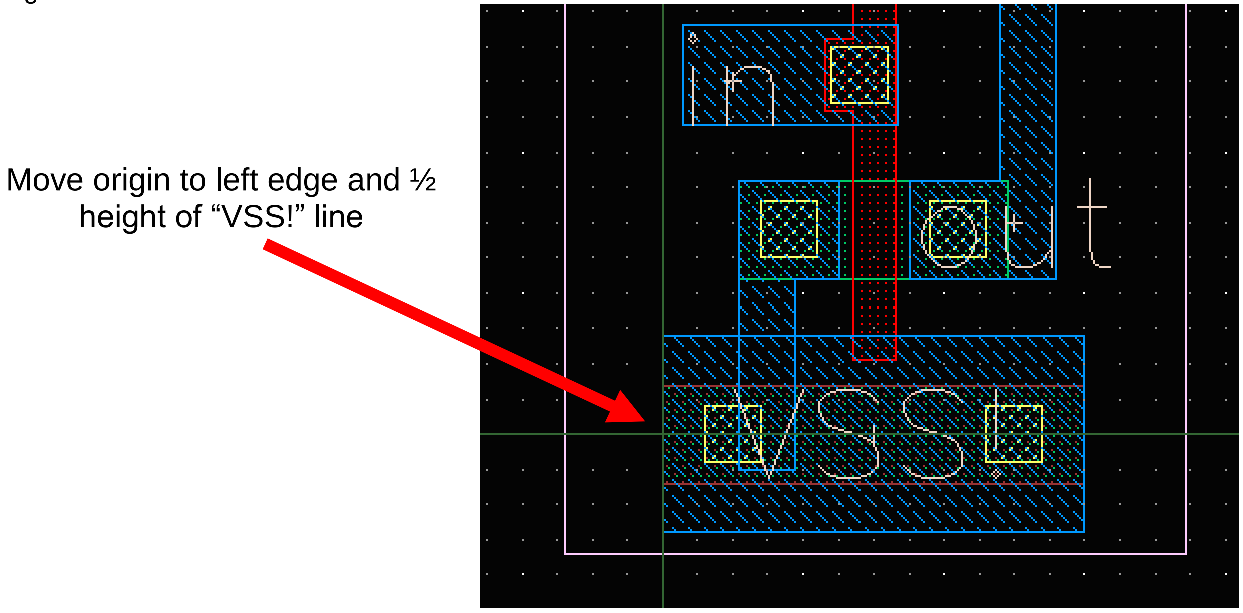
Fig. Important design rules for IBM 013 um CMOS Technology

After you complete layout of the inverter, you need to draw "GRLOGIC" layer to enclose your inverter; this is to ensure your design is exempt from stringent design rule check such as latchup checks. You will also need to create labels for your I/O and power nets. First select the metal of your I/O or power net, and then select ***Create/Label…*** (Shortcut: 'l'). The "Create Label" window will appear; enter the net name for Label (in, out, VDD!, or VSS!), and change the height to 0.2. "Justification" can be changed to any position; when you see the label being created in layout view, you will find a little "+" sign along with your label. This "+" sign is the Justification; **please note that Justification has to be on top of your metal layer**. After the label is created, click on the label name and press 'q' (for editing properties), and make sure that Layer name is "M1 – ll" (indicating that it's a label). Alternatively, you can select the "M1 - ll" layer in the LSW window before creating the label.

<u>The following parts (grid, origin, pitch) are very important to your future CAD assignments</u>

Label Grid
In order to ease eventual automatic routing between layout cells, you need to ensure that your label “Justification” (X and Y origin) is located on a predefined grid. In IBM 0.13 μm technology, this grid is located in 0.4 μm intervals with a 0.2 μm offset at the boundaries of the cell (for metal layers 1-6, see the IBM documentation for the details on metals 7 and 8). This means that (starting from the origin), appropriate X grid label origins are **0.2, 0.6, 1.0, 1.4**, … and appropriate Y grid label origins **are 0.2, 0.6, 1.0, 1.4**, etc. These grid spacing’s are enforced within the design rule checks, so make sure you adhere to this grid, otherwise your designs will not be DRC clean.

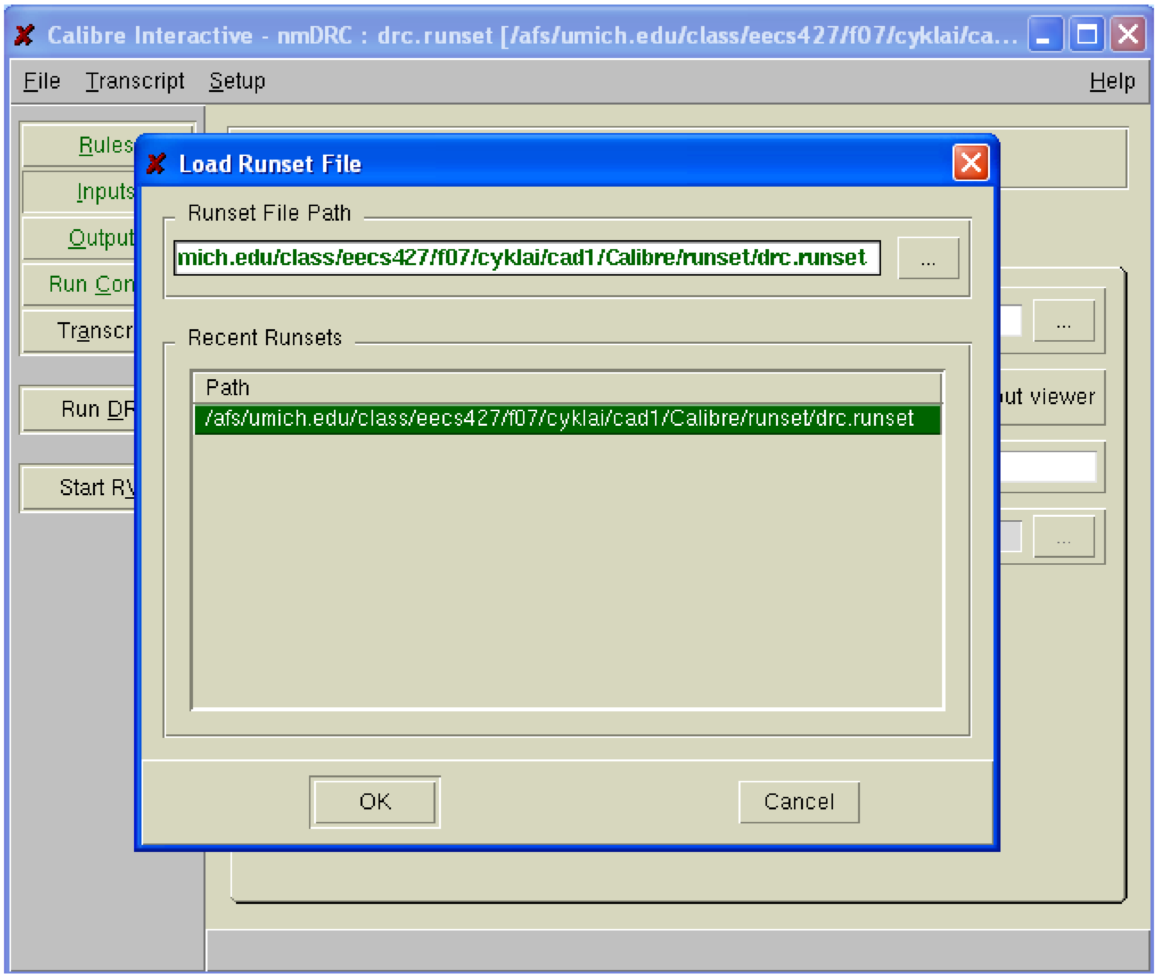
Setting the Cell Origin
Setting a cell’s origin at the correct location makes reading cell coordinates easier (especially when placing labels on the correct grid, as described above). Ideally, you’d like to set the cell’s origin to the left edge of the Metal1 “VSS” line, halfway between the vertical edges (i.e., if the Metal1 VSS line is 0.56 μm tall, place the origin halfway, 0.28 μm into the metal line). The figure below illustrates this concept. Select ***Edit->Other->Move Origin*** and click to set the origin at a new location.



Pitch Matching
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.

H. Design Rule Checking (DRC)

After layout is completed, you need to perform a Design Rule Check (DRC) to verify that your layout meets all design rules. First select ***Calibre->Run DRC*** in the Layout Editing window. The DRC form with the name “Load Runset File” appears; please select the path for the DRC runset file. The runset file is stored under ***cad1/Calibre/runset/drc.runset***. Select the runset file and then click OK. The image of the DRC window is shown below:


Calibre Interactive - nmDRC : drc.runset [/afs/umich.edu/class/eecs427/f07/cyklai/ca...

Load Runset File

Runset File Path: mich.edu/class/eecs427/f07/cyklai/cad1/Calibre/runset/drc.runset

Recent Runsets

Path
/afs/umich.edu/class/eecs427/f07/cyklai/cad1/Calibre/runset/drc.runset

Fig. DRC runset window

After you set the DRC runset file, click “Run DRC” on the left panel of the “nmDRC” window. A couple of windows will be generated once DRC completes. If you look into the “DRC RVE” window, you should see a list of design rules. **If your DRC is clean, you will see green check marks beside all the design rules; if not, you will see red marks beside the rules**. To debug DRC errors, you can right-click on the design rules and select “Highlight Cluster”, and it will highlight the error in your layout. Once you fix your layout, save it and then click on “Run DRC” to rerun DRC. Examples of clean DRC and dirty DRC RVE window are shown below:

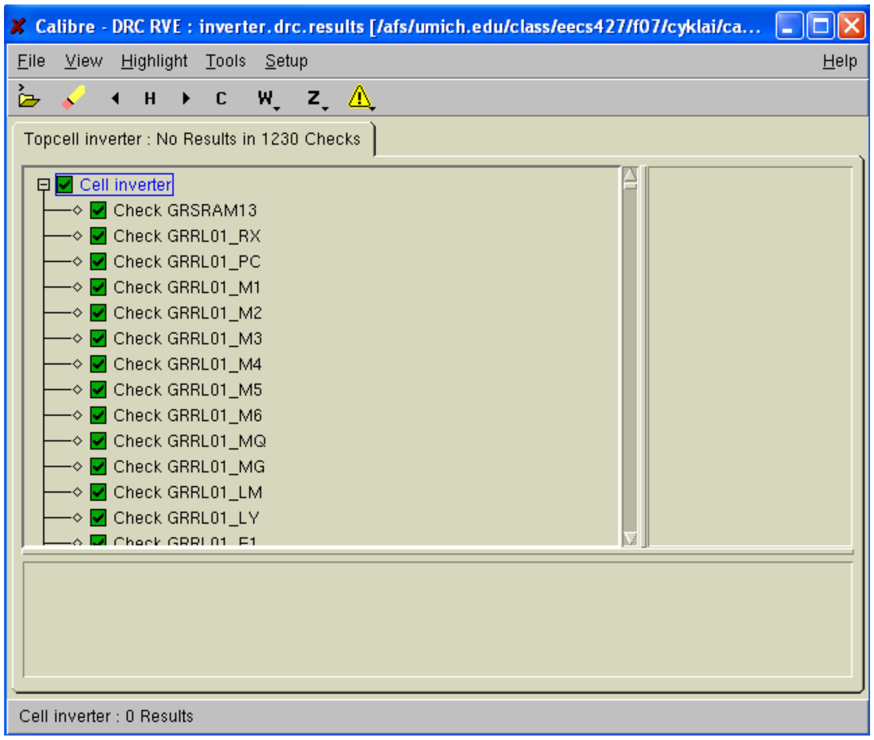
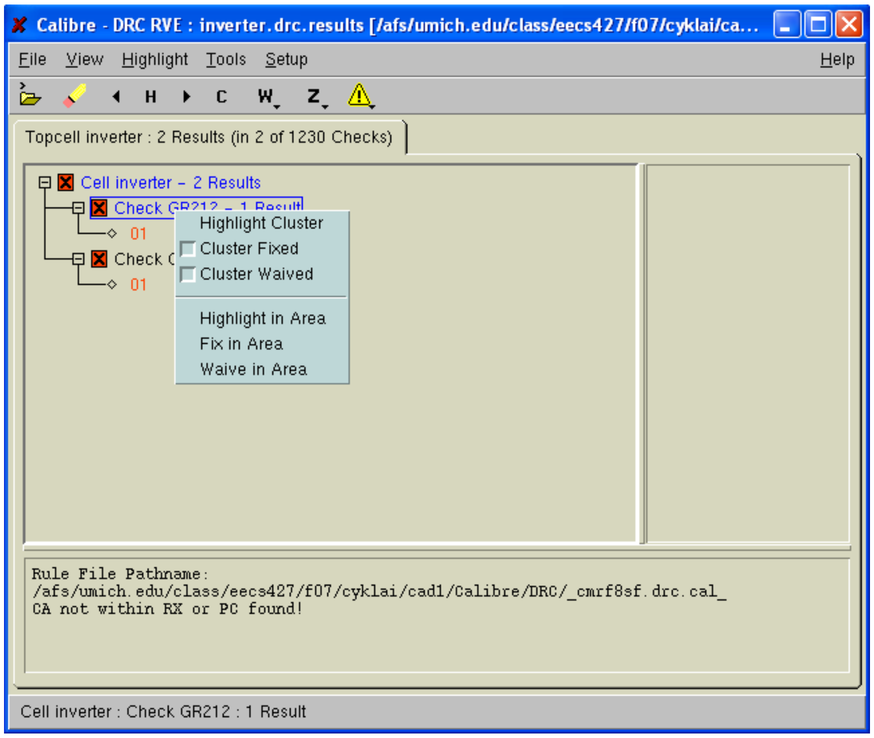


Fig. Clean DRC Window


Calibre - DRC RVE : inverter.drc.results [/afs/umich.edu/class/eecs427/f07/cyklai/ca...
File View Highlight Tools Setup Help
H C W Z
Topcell inverter : 2 Results (in 2 of 1230 Checks)
Cell inverter - 2 Results
Check GR212 - 1 Result
01
Check
01
Highlight Cluster
Cluster Fixed
Cluster Waived
Highlight in Area
Fix in Area
Waive in Area
Cell inverter : Check GR212 : 1 Result

```
Rule File Pathname:
/afs/umich.edu/class/eecs427/f07/cyklai/cad1/Calibre/DRC/_cmrf8sf.drc.cal_
CA not within RX or PC found!
```

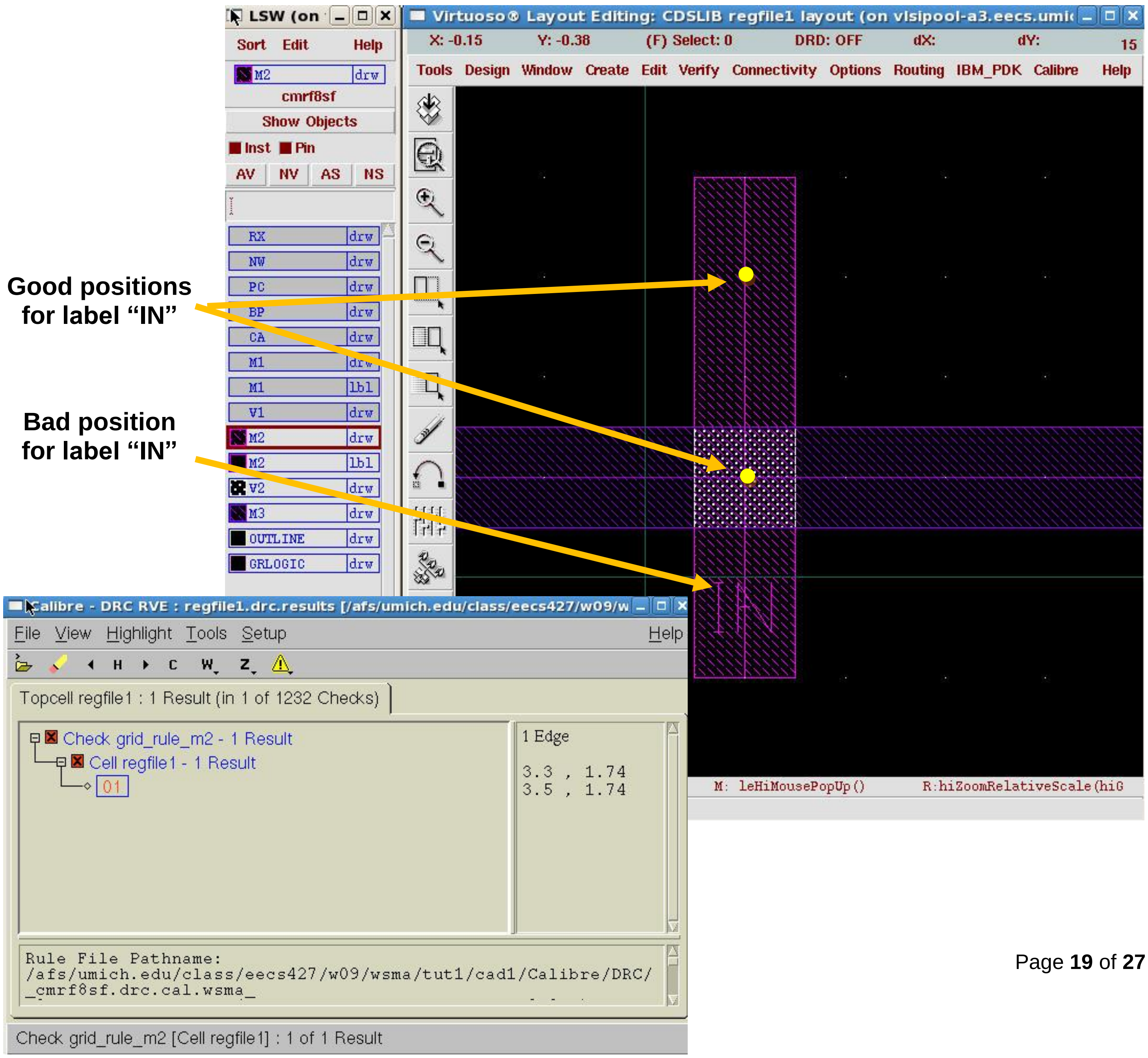
Fig. Dirty DRC Window

To learn more about each design-rule, you can look it up in the design rules provided in the class directory: ***/afs/umich.edu/class/eecs427/ibm13/docs/layout_rules_427.pdf***. Note the rule number will correlate to the one in the handout. 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".

DRC report can be found under the directory ***cad1/Calibre/DRC/inverter.drc.results*** and ***cad1/Calibre/DRC/inverter.drc.summary***.

Side Note: Metal-Label-Layer Grid check

The DRC rules have been modified to enforce the metal-label-layer grid rules established for IBM 0.13um (described previously on page 15). Following these grid rules greatly improves the automatic routing efficiency (which will be important later in the course). The figure below illustrates what the metal grid DRC errors will look like. Note that the misalignment error bars are actually offset from the labels by 0.2um for metals 1-6 and 0.4um for metals 7 and 8. Vertical grid errors appear as horizontal bars and horizontal errors appear as vertical bars.


Good positions for label "IN"
Bad position for label "IN"
LSW
Sort Edit Help
M2 drw
cmrf8sf
Show Objects
Inst Pin
AV NV AS NS
RX drw
NW drw
PC drw
BP drw
CA drw
M1 drw
M1 lbl
V1 drw
M2 drw
M2 lbl
V2 drw
M3 drw
OUTLINE drw
GRLOGIC drw
Virtuoso® Layout Editing: CDSLIB regfile1 layout (on vlsipool-a3.eecs.umi
X: -0.15 Y: -0.38 (F) Select: 0 DRD: OFF dX: dY: 15
Tools Design Window Create Edit Verify Connectivity Options Routing IBM_PDK Calibre Help
IN
M: leHiMousePopUp() R:hiZoomRelativeScale(hiG
Calibre - DRC RVE : regfile1.drc.results [/afs/umich.edu/class/eecs427/w09/w
File View Highlight Tools Setup Help
H C W Z

Topcell regfile1 : 1 Result (in 1 of 1232 Checks)

Check grid_rule_m2 - 1 Result
Cell regfile1 - 1 Result
01

```
1 Edge

3.3 , 1.74
3.5 , 1.74
```

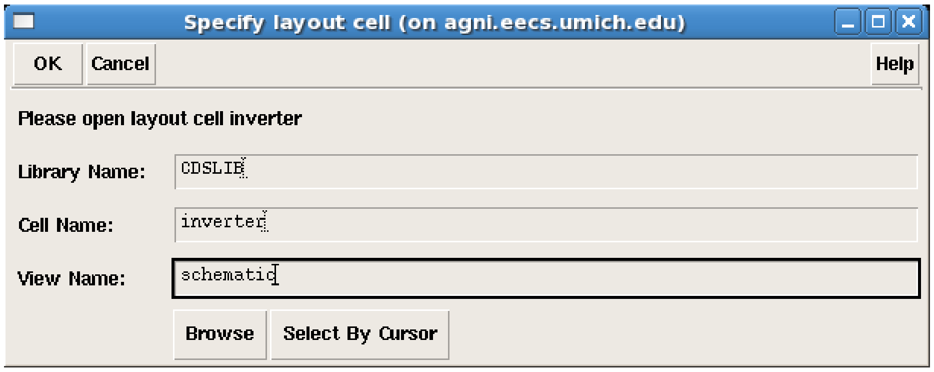
```
Rule File Pathname:
/afs/umich.edu/class/eecs427/w09/wsma/tut1/cad1/Calibre/DRC/
_cmrf8sf.drc.cal.wsma_
```

Check grid_rule_m2 [Cell regfile1] : 1 of 1 Result

I. Layout vs. Schematic (LVS)

When you are done with DRC for your layout, save the design, and then select ***Calibre->Run LVS*** in the Layout Editing window**.** Load the LVS runset file in the same way as DRC; this time remember to use ***cad1/Calibre/runset/lvs.runset*** as the LVS runset file. Then click OK.

After runset file is loaded, press "Run LVS" on the left panel of the LVS window. A window named "Specify layout cell" will appear; **remember to change the view name from "layout" to "schematic"**, shown below. Press OK.



Please open layout cell inverter

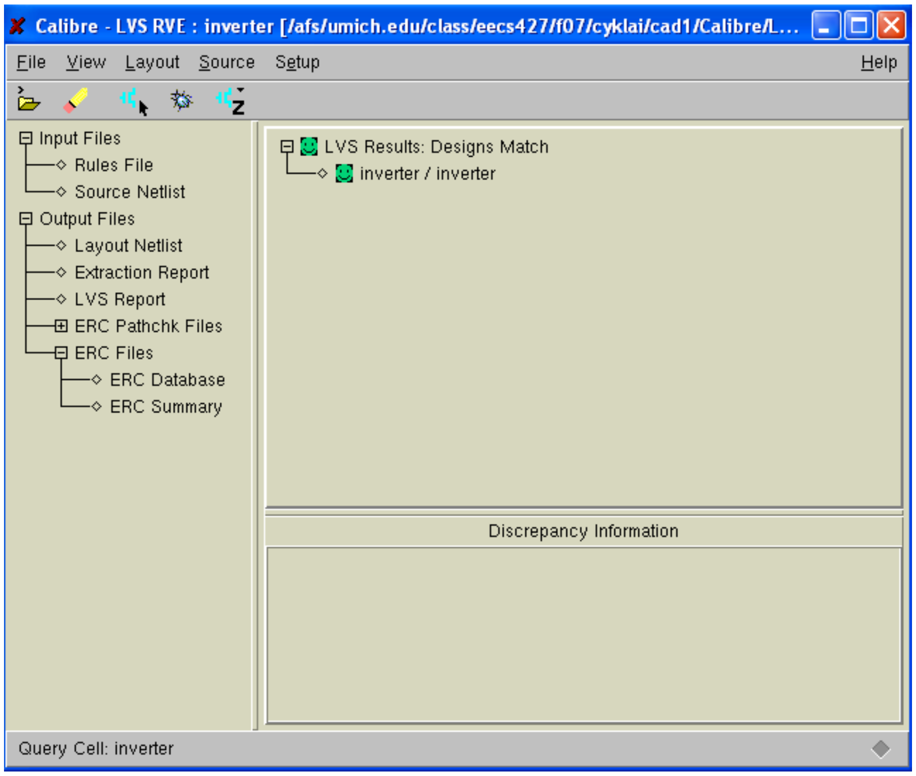
Library Name: CDSLIB

Cell Name: inverter

View Name: schematic

Fig. Specify layout cell window

After LVS completes, a couple of windows will appear. If your LVS is clean, you will see a green smiley face in your LVS RVE window, as shown below:

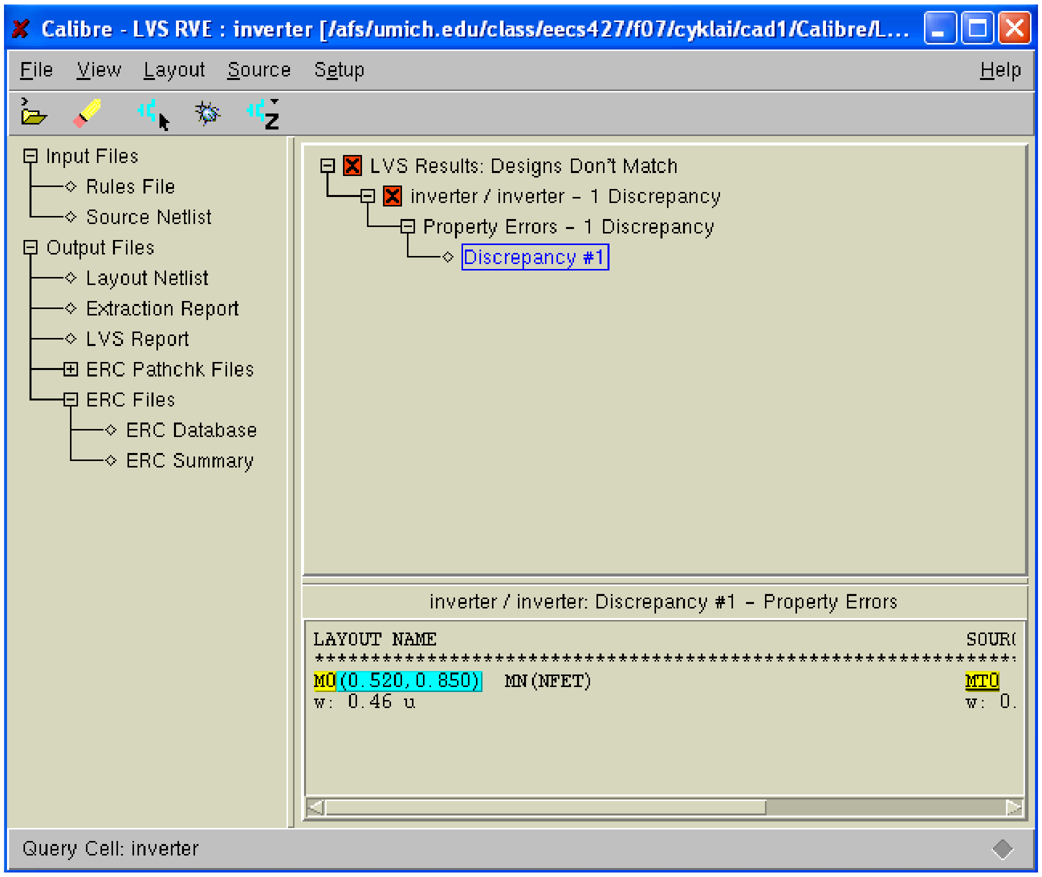


LVS Results: Designs Match
inverter / inverter

Query Cell: inverter

Fig. Clean LVS Window

Otherwise you will see red marks in the LVS RVE window, as shown below. In order to debug LVS errors, you can click on "Discrepancy", and RVE will show you the detail of the LVS error on the bottom panel. You can double-click on the device instance name (for the example below, it's M0), and the respective area in layout will be highlighted just like DRC errors. Fix all the errors, save the layout, and then rerun LVS until LVS is clean.


Calibre - LVS RVE : inverter [/afs/umich.edu/class/eecs427/f07/cyklai/cad1/Calibre/L...
File View Layout Source Setup Help
Input Files
Rules File
Source Netlist
Output Files
Layout Netlist
Extraction Report
LVS Report
ERC Pathchk Files
ERC Files
ERC Database
ERC Summary
LVS Results: Designs Don't Match
inverter / inverter - 1 Discrepancy
Property Errors - 1 Discrepancy
Discrepancy #1
inverter / inverter: Discrepancy #1 - Property Errors
Query Cell: inverter

```
LAYOUT NAME                                                    SOUR(
*******************************************************************:
M0(0.520,0.850)   MN(NFET)                                     MT0
w: 0.46 u                                                      w: 0.
```

Fig. Dirty LVS Window

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 VSS 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 VSS 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. The LVS report can be found under the directory ***cad1/Calibre/LVS/inverter.lvs.report***.

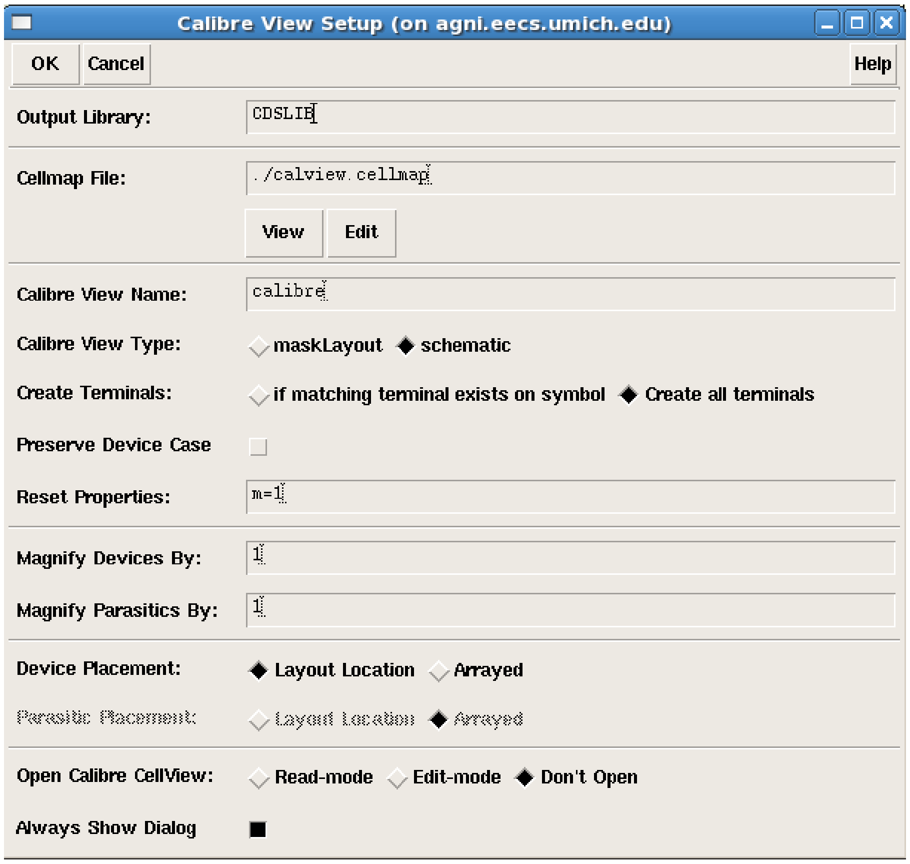
After you get LVS clean, you may want to make little changes to purposely cause LVS failures to see the error message. This will help you in your future designs. Here's a brief list of things that you can try:

1. Shorted power and ground
2. Shorted signal nets
3. Open signal nets
4. Change device sizes
5. Missing N-well or substrate contact
6. Pin name mismatch

J. Layout Parameter Extraction

Now we're going to extract the wire capacitances from the layout. Select ***Calibre->Run PEX*** in the Layout Editing window**.** Load the runset file in the same way as DRC and LVS; this time remember to use ***cad1/Calibre/runset/xRC.runset*** as the PEX runset file. Then click OK. Press "Run PEX" on the left panel of the PEX window.

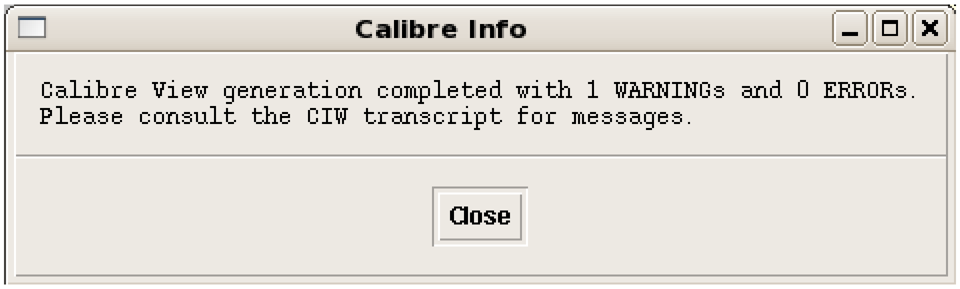
After PEX is completed, a couple of windows will appear. In the "Calibre View Setup" window, change "Calibre View Type" to "schematic"; change "Create Terminals" to "Create all terminals". Your window should look like this:



Output Library: CDSLIB
Cellmap File: ./calview.cellmap
Calibre View Name: calibre
Calibre View Type: maskLayout [✓ schematic]
Create Terminals: if matching terminal exists on symbol [✓ Create all terminals]
Preserve Device Case ☐
Reset Properties: m=1
Magnify Devices By: 1
Magnify Parasitics By: 1
Device Placement: [✓ Layout Location] Arrayed
Parasitic Placement: Layout Location [✓ Arrayed]
Open Calibre CellView: Read-mode Edit-mode [✓ Don't Open]
Always Show Dialog ■

Fig. Calibre View Setup

Press OK. A new Window should pop-up and tell you have 1 Warning and 0 Error. Press OK.



```
Calibre View generation completed with 1 WARNINGs and 0 ERRORs.
Please consult the CIW transcript for messages.
```

Fig. Calibre pop-up Window

Go back to the Library Manager, and open the “calibre” View of your “inverter” cell. A window named “Virtuoso Schematic Reading” should appear showing a PMOS, an NMOS, a diode, and a few capacitors. This is the post-layout extracted view of the schematic. Now you can move on to post-extraction simulation.

K. Simulation of Extracted Netlist

Now that we have extracted the parasitic, 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 of the “inverter_test,” and start the Analog Design Environment. Once you are inside click ***Setup->Environment***. In the “Switch View List” make sure “calibre” exists; if it doesn’t, add “calibre” just before “schematic”; this step is to ensure that HSpice will fetch the post-layout parasitic. Now you can run the HSpice simulation as before to measure the values.

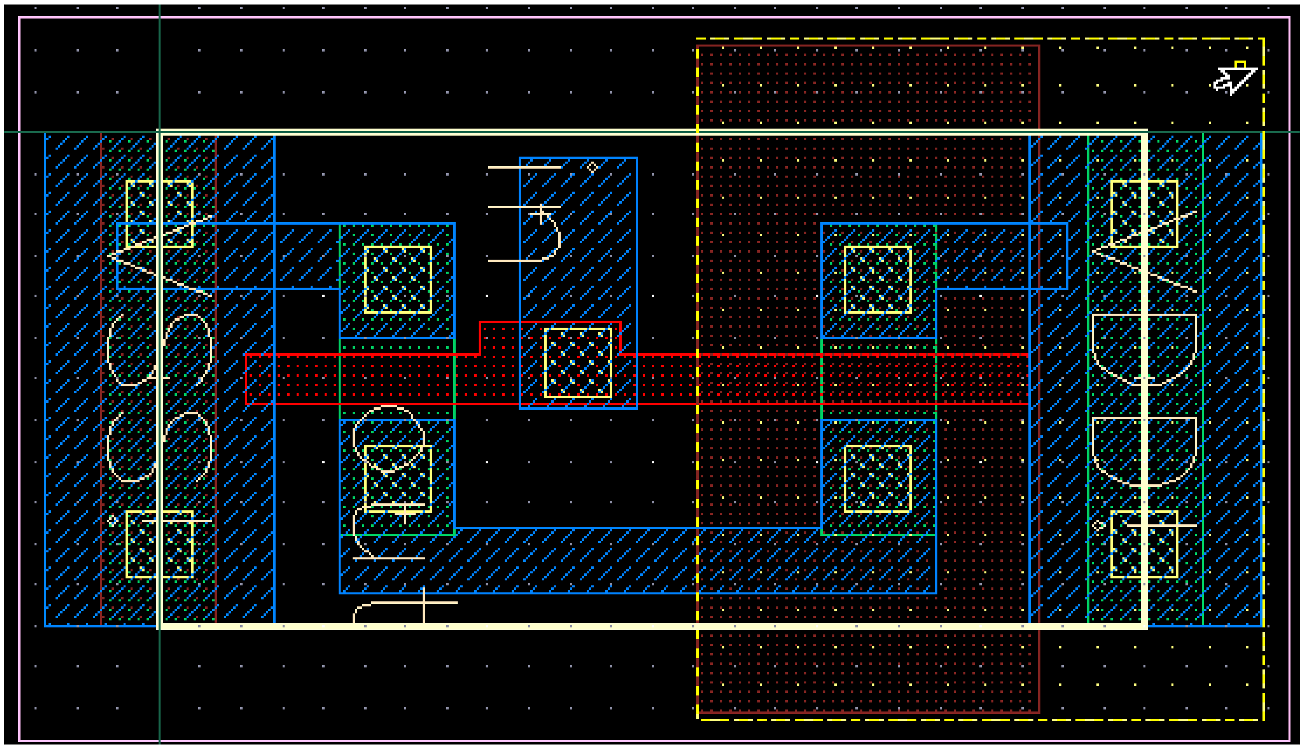
L. Digital Delay Annotation and Resimulation

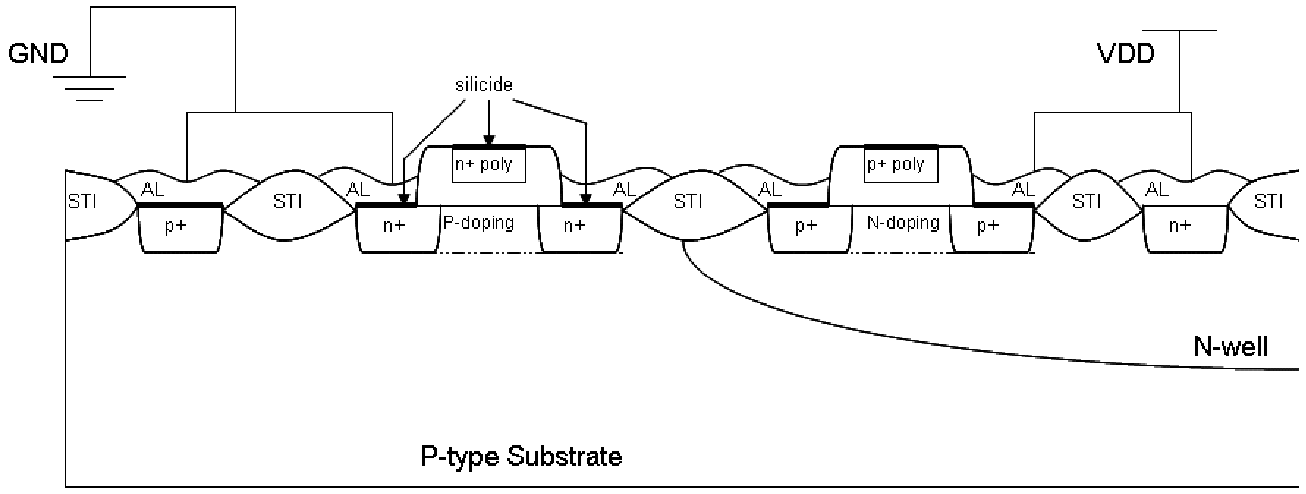
Since the logic is so simple, we will give you the instruction later when you are doing a larger design.

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 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: Cross view of an inverter





http://www.eeinterview.com/

Appendix B: 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 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 Cadence. 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 of 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 (Standard Library Cell Names)
dffsx[1,2,4,8] (Shorthand for dffsx1, dffsx2, dffsx4, and dffsx8)
fill[1,2,3]
invx[1,2,4,8]
mux2x[1,2,4,8]
nor2x[1,2,4,8]
tielo
xorx[1,2,4,8]
dffrx[1,2,4,8]
nand2x[1,2,4,8]
tiehi
tristatex[1,2,4,8]

Appendix C: Hierarchical Layout Design

For the inverter, nothing in this paragraph needs to be done, but refer back to it for other CAD assignments. 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 (Shortcut: 'i'). 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 D: 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.