ELEC 422/527
VLSI Systems Design - Spring 2019
Notes on Verilog / Design Compiler / Encounter / Magic / Ext2sim / Irsim flow
AND Gate Example

The template folder for this process is provided on the ssh.clear.rice.edu cluster in the elec422 account. This set of notes on the flow will start by copying the files and running some scripts along with interactive commands. These notes assume that the start-up files for the bash or csh shell from the elec422 account have been installed and that the X windows display is set up.

(Note that cutting and pasting from this document may not work properly since Microsoft Word often substitutes other characters for "/" or "_" that may lead to errors in Linux.)

1. The template files are located at: /clear/courses/elec422/2019_spring/AND_template. Also, please read the README files that are in the template folders. Some parts are included here too. Begin by copying this whole folder and subfolders to your local folder. Choose your own name for the folder. I am creating ASIC_built for this example as a sub-folder to my current demo folder indicated by the "~/demo" here. (There were additional hidden configuration files starting with a "." that are not needed for your new projects.

```
cp -r ~elec422/2019_spring/AND_template ~/demo/AND_built
```

There are multiple folders for each step of the design process which you can see by typing "ls"

```
[elec422@amber AND_built]$ pwd

/clear/courses/elec422/demo/AND_built

[elec422@amber AND_built]$ ls

Encounter  Hspice  Irsim  Modelsim  Modelsim_post_DC  Synthesis
```

Not that there is a pre design compiler Modelsim folder and a separate Modelsim_post_DC folder. You will need to copy some of your files from one folder to the next when your build your own homework and project designs. It is "safer" to use separate folders to prevent the overwriting of important files.

1a. Modelsim before synthesis. We will also do Modelsim simulation before and after. Make sure to change directory to the Modelsim folder.

Creating a library directory:

ModelSim requires a “library directory” to store information about your project. In your Modelsim directory (where the simulation project .v files are), type

```
vlib work
```

This creates a sub-directory called work needed by Modelsim. If you delete this directory, you will need to create it again before running any compilations.

Compiling Verilog files:

Next from the directory where your Verilog design and testbench files are, type

```
vlog «list of verilog files»
```

This might look like

```
vlog my_and.v testbench.v
```

Or, if all of the .v files are pertinent, just type

```
vlog *.v
```

This compiles your .v files for simulation. If there were no errors, the compilation should list the modules compiled and point out the top level modules (ones not included by any other modules). If the compilation failed with ERR messages, you will need to revise your .v files to address the complaints and recompile. It is a good practice to also look for WARN messages and resolve those as well.

Running ModelSim: To run ModelSim, from the working directory, type:

```
vsim -voptargs="+acc" «top-level module name»
```

For example,

```
vsim -voptargs="+acc" my_and_tb
```

if my_and_tb is the top level testbench module in your design. In an X-ready environment, this should bring up the simulator main window.

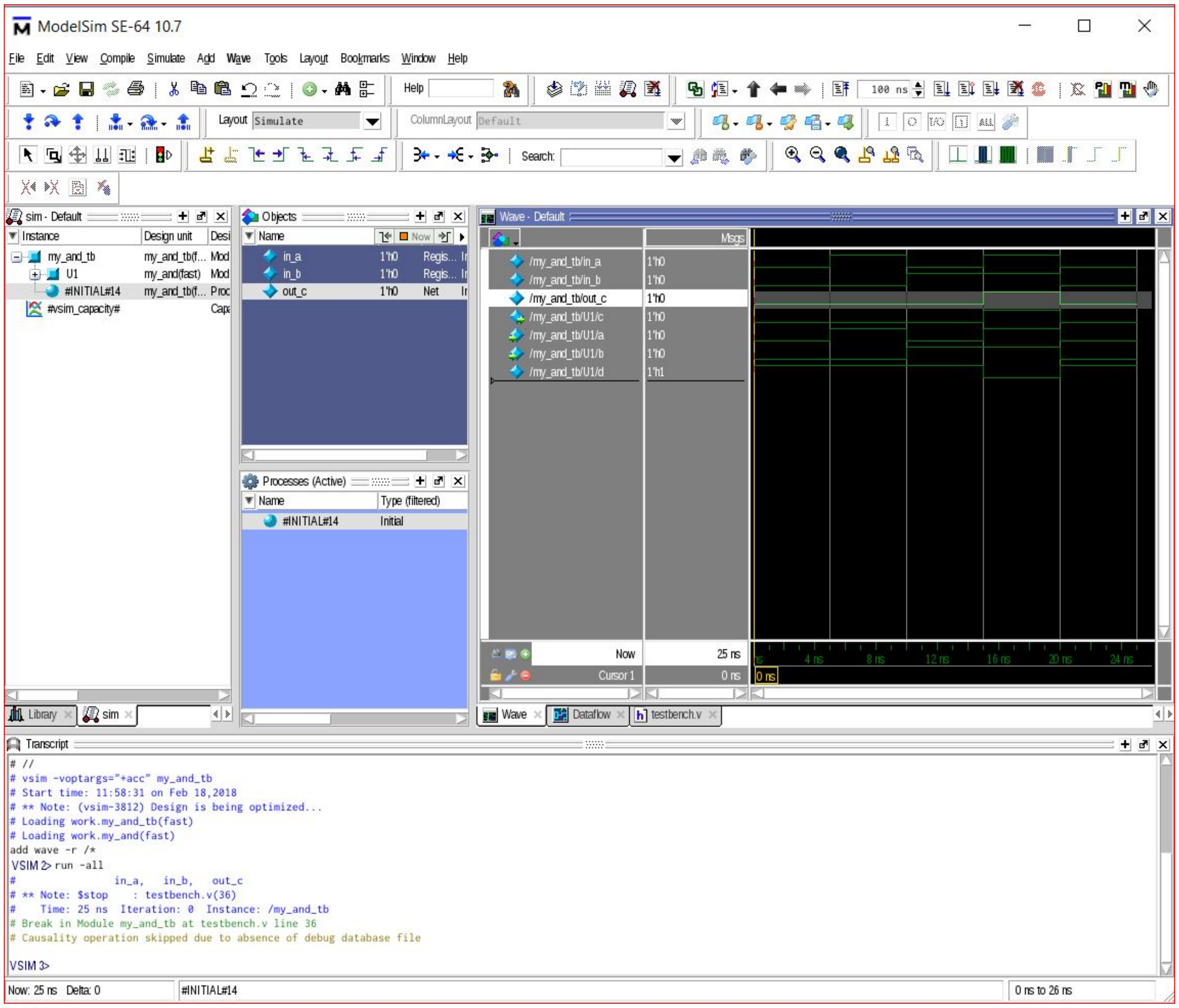
The -voptargs="+acc" helps to see all signals. You will need to right click in the "Objects" block to "Add -> To Wave -> Signals in Design" to see all waveforms in the wave window.

You can start the simulation by typing in the simulator bottom command window:
run -all

The simulation will run until it encounters a $stop or $finish command in your .v files.

In the Wave window, then you can click on the tab at the bottom that says "Wave" to bring that window on top. You can right click in the waveform window to "Zoom Full" or use the slider to navigate the simulation.

In the lower left of the Wave window there are six icons. On the top row, there is one that that "toggles leaf names." This helps to show or hide the Verilog sub-cell hierarchy. Also, to the right of that icon, there is "Grid, Timeline, and Cursor Control" button that pops up a menu. On the Display tab of the pop-up menu, you may want to select "Waveform selection highlighting." This can help you since now clicking on a signal in the Wave window will not highlight that signal.



Instance	Design unit	Desi
my_and_tb	my_and_tb(f...	Mod
U1	my_and(fast)	Mod
#INITIAL#14	my_and_tb(f...	Proc
#vsim_capacity#		Capa

Name	Now	
in_a	1'h0	Regis...
in_b	1'h0	Regis...
out_c	1'h0	Net

Name	Type (filtered)
#INITIAL#14	Initial

	Msgs
/my_and_tb/in_a	1'h0
/my_and_tb/in_b	1'h0
/my_and_tb/out_c	1'h0
/my_and_tb/U1/c	1'h0
/my_and_tb/U1/a	1'h0
/my_and_tb/U1/b	1'h0
/my_and_tb/U1/d	1'h1
Now	25 ns
Cursor 1	0 ns

```
# //
# vsim -voptargs="+acc" my_and_tb
# Start time: 11:58:31 on Feb 18,2018
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.my_and_tb(fast)
# Loading work.my_and(fast)
add wave -r /*
VSIM 2> run -all
#                in_a,   in_b,   out_c
# ** Note: $stop    : testbench.v(36)
#    Time: 25 ns  Iteration: 0  Instance: /my_and_tb
# Break in Module my_and_tb at testbench.v line 36
# Causality operation skipped due to absence of debug database file

VSIM 3>
```

Now: 25 ns Delta: 0 | #INITIAL#14 | 0 ns to 26 ns

2. There will be a Synthesis and an Encounter folder. First change to the Synthesis folder with:

```
cd Synthesis
```

There will be 3 files here, the run and .tcl file along with your personal Verilog file or files for a larger design:

```
compile_dc.tcl  my_and.v  run
```

The tcl file has been modified to list the my_and files. You will need to change this for any new original design that you make. (Note that you do not include the testbench file.) Use a text editor such as "vi" or "gedit" to edit the .tcl file. You would need to change the list of Verilog files, the top level module, the clock pin name, and the target clock frequency for a new design. You would replace the my_and.v file with your own Verilog files for the homework and project.

3. We can execute Synopsys design compiler now. The script "run" contains the appropriate commands to start design compiler in non-graphical mode and read in the .tcl command file. So just type:

```
run
```

3a. Modelsim after Synthesis.

This readme file is after the synthesis of the Verilog files. It differs slightly from the PRE_DC process in several ways:

1. The output Verilog file, here with a .vh extension is used
2. A Verilog file for the cells in the standard cell library needs to be compiled in as well.

~~((Note: although mentioned in class the timescale command may not be needed in the testbench in the new version of Modelsim after synthesis.~~
~~3. The testbench.v file needs an extra definition in the first line exactly as: `timescale 1ns/10ps))~~

In the template and built folders on CLEAR, there are Modelsim_post_DC folders which are separate of the before design compiler Modelsim folder. This is to help preserve the results before and after Design Compiler. The mapping to actual logic gates and flip-flops often makes subtle changes in the simulation results.

Creating a library directory

ModelSim requires a “library directory” to store information about your project. In your directory (where the simulation project .v files are) Modelsim_post_DC, type

```
vlib work
```

This creates a sub-directory called work needed by Modelsim. If you delete this directory, you will need to create it again before running any compilations.

Compiling Verilog files

Next from the working directory, type

```
vlog «list of verilog files»
```

This might look like after Design Compiler synthesis as:

```
vlog my_and.vh osu05_stdcells.v testbench.v
```

This compiles your .v (and synthesized output .vh) files for simulation. If there were no errors, the compilation should list the modules compiled and point out the top level modules (ones not included by any other modules). If the compilation failed with ERR messages, you will need to revise your .v files to address the complaints and recompile. It is a good practice to also look for WARN messages and resolve those as well. Note that the order of files on the vlog command can affect the timing results. It is suggested to have the design Verilog files first, then the osu05_stdcells.v next, and the testbench.v last.

Running ModelSim

To run ModelSim, from the working directory, type:

```
vsim -voptargs="+acc" «top-level module name»
```

For example,

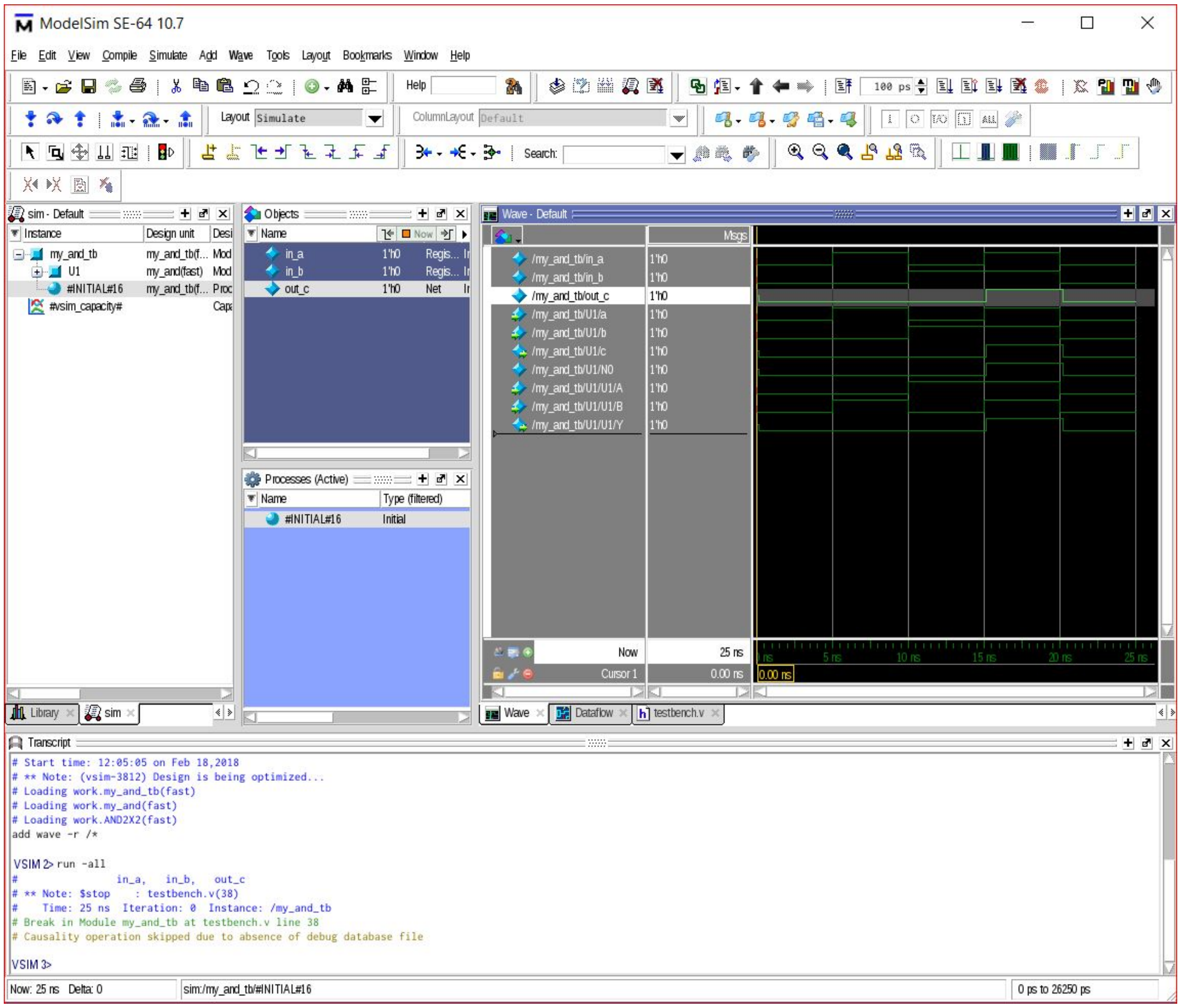
```
vsim -voptargs="+acc" my_and_tb
```

if my_and_tb is the top level testbench module in your design. In an X-ready environment, this should bring up the simulator main window.

The -voptargs="+acc" helps to see all signals. You will need to right click in the "Objects" block to
"Add -> To Wave -> Signals in Design" to see all waveforms in the wave window.

You can start the simulation by typing in the simulator command window:
run -all

The simulation runs until it encounters a $stop or $finish command in your .v files.



```
# Start time: 12:05:05 on Feb 18,2018
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.my_and_tb(fast)
# Loading work.my_and(fast)
# Loading work.AND2X2(fast)
add wave -r /*

VSIM 2> run -all
#                in_a,   in_b,   out_c
# ** Note: $stop    : testbench.v(38)
#    Time: 25 ns  Iteration: 0  Instance: /my_and_tb
# Break in Module my_and_tb at testbench.v line 38
# Causality operation skipped due to absence of debug database file

VSIM 3>
```

Now: 25 ns Delta: 0   sim:/my_and_tb/#INITIAL#16   0 ps to 26250 ps

4. Copy output files for placement and routing. When design compiler completes, then you will need to copy the new Verilog file that has the structural netlist and the names of the cells in our AMI05 library along with the timing information file to the Encounter folder. In this example, the files start with my_and which will of course change to your own Verilog file names.

```
cp my_and.vh ../Encounter/
```

`cp my_and.sdc ../Encounter/`   *(Copying the .sdc file is optional since we are not optimizing timing.)*

5. Change to the Encounter folder for the remainder of the design flow and look at the other template files there. The .tcl, and the run file are always needed.

```
cd ../Encounter
```

You will see: `encounter.tcl` and `run`

encounter.tcl has been edited and modified to list the top level cells as:

```
set my_toplevel  my_and
```

You will need to change this for any new design name to match the top level cell name.

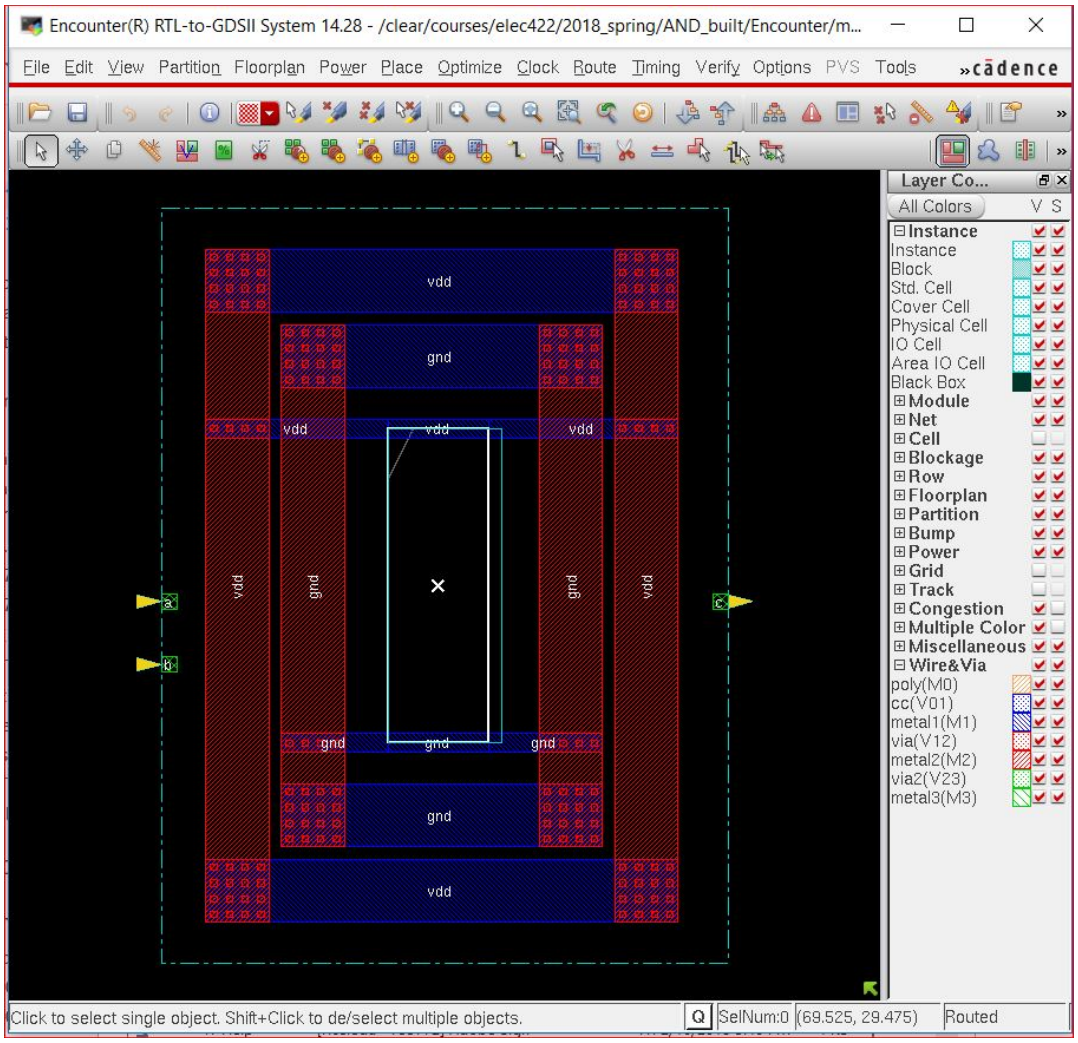
6. Now we can use the Encounter folder "run" script to build the layout and the GDS II mask layout file. We will be using Encounter version 14 and the newest version of the scripts will be in /clear/courses/elec422/scripts_new/Encounter. I will post an announcement about any changes or updates that are needed for reliable layout generation. You should then copy and use the update suggested. The `encounter.tcl` file will always call this `final.gds` independent of the main Verilog file names. (It could be changed, but it is simpler to have a standard name at this point since it is a placeholder file for the entire design.) After typing

run

and waiting for a while, the layout will be created. You can type "win" to pop up a layout window and then "exit" to quit.

There may be several warnings concerning features not included in our cell library. These will be saved in a log file. Most can be ignored for our designs. If there is an error, it may be due to not updating your verilog files or the requested aggressive cell density 0.9 in the floorplan may be too high.

Encounter will go through several stages to (1) load and initialize the design, (2) floorplan your design, (3) add power and ground rings, (4) place the design, (5) route power and ground, (6) add filler cells to rows, (7) detail signal route, (8) verify design, and (9) create the GDS layout mask file. More steps could be done for larger designs including timing optimization, clock tree routing, and also low power modes.



Now that we have a GDS II layout file, we will need to extract and convert this to a simulator input file for the irsim switch level simulator through the `magic` and `ext2sim` programs.

7. Start the magic VLSI editor by typing:

```
magic
```

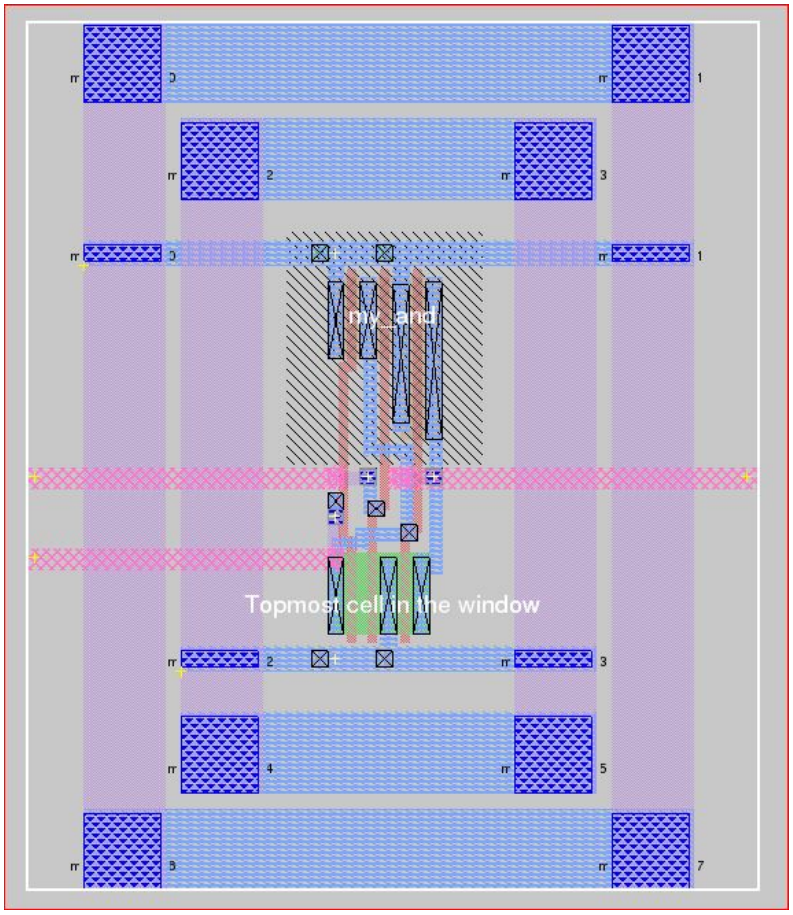
Two windows should pop up again through the X windows environment, one is a tcl/tk command console window (tkcon) where certain text commands are typed, and the other is the layout window which has its own title bar with commands. We will use both and for different steps. There are several tutorials on the magic layout editor online. We are using Version 8.1 now and tutorials are at: http://opencircuitdesign.com/magic/magic_docs.html From the magic layout window, click in the layout window and then type the long command:

: calma rescale no

This helps to deal with some internal grid scaling issues between Encounter and Magic. Magic will show poly as 4 grid boxes instead of the normal 2 grid boxes if rescaling is not disabled which can be visually confusing. Now we can load the gds files as:

`File - > Read GDS ->` and then select the file `final.gds`

The layout view should look similar as in the last display step for Encounter in the notes above.



Some notes on starting magic based on version 7 are in tutorial 1 at:

http://opencircuitdesign.com/magic/archive/papers/tut1.pdf

We would now like to verify that all of the sub-cell layout has been included in our final.gds file. To do this, we are going to put a "box" around the entire design and then expand the sub-cells.

To better understand selection of cells and moving and zooming around the layout in magic you can refer to the second magic tutorial at:
http://opencircuitdesign.com/magic/archive/papers/tut2.pdf

Move your mouse cursor to the layout window and position it below and left of your design. Click the left mouse button for the lower left corner of bounding box. Now, move the mouse above and to the right of your entire design and click the right mouse button for the upper right corner of bounding box. There should be a white outline around your entire layout. Try to reposition the bounding box again if it does not surround the entire design. Now expand the layout with the ***"long command" which starts with a colon.*** (Note that you keep the mouse in the layout window when you do this. You do not click and focus on the tkcon console window. However, when you do type in the layout window, the text that you type appears in the console window.)

```
:expand
```

You should inspect the design for connectivity issues that can occur. After expanding the design, you can "select" with the "s" short command (no colon) in the layout window to see wire connectivity. For example, the "gnd" label turns white to indicate that it is properly connected to the wire. Do this by placing the mouse cursor over the metal line before you type "s". You can resize and move the bounding box there is you wish, but clicking is not really necessary. Each time that you type "s" magic looks further in your layout, that is first the current box, then the boxes connected to the current box, and eventually back to the first current box. You may need to be patient in a large design as it takes a few seconds for magic to trace through contacts and layer changes.

8. Save the magic layout files. Now from the top menu in the magic layout window select

```
File -> Save layout.
```

When the dialog box pops up, we want to save all of the sub-cell files as well, so select:

```
autowrite
```

There will be a warning about a cell "UNNAMED" but we can ignore that, since it corresponds to a top level cell that we can ignore.

9. First step of the extraction process. Magic will now find the transistors, contact, wires, and estimate the resistance and capacitance of the node in your design. Encounter has done something similar but not in a format for the irsim simulator. So, now type the "long command" in magic:

```
: extract all
```

10. Second step to combine .ext files into a single .sim file. We need to now turn the multiple extractor files from magic into a single simulator .sim file to be the input for irsim. You can do this from within magic with:

```
: ext2sim -t!
```

This will create a .sim file for the top level cell which in this case is my_and.sim

Also in some cases where we will do Hspice simulation, run:

: ext2spice

We have now completed the parts in the magic editor. You can now leave magic and close both windows from the command in the layout window of

```
File -> Quit
```

11. Simulation with irsim. Now we can simulate the design with irsim.

There should be a sample irsim command file in the Irsim folder. The .cmd file is a demo template for the AND gate for Irsim running on Linux which understand the "V" commands You will need to copy the .sim file from the Encounter folder to the Irsim folder.

If you are currently still in the Encounter folder, (check by doing a "pwd" to print the current working directory), then

```
cp my_and.sim   ../Irsim
```

Now you change directory to the Irsim folder with:

```
cd ../Irsim
```

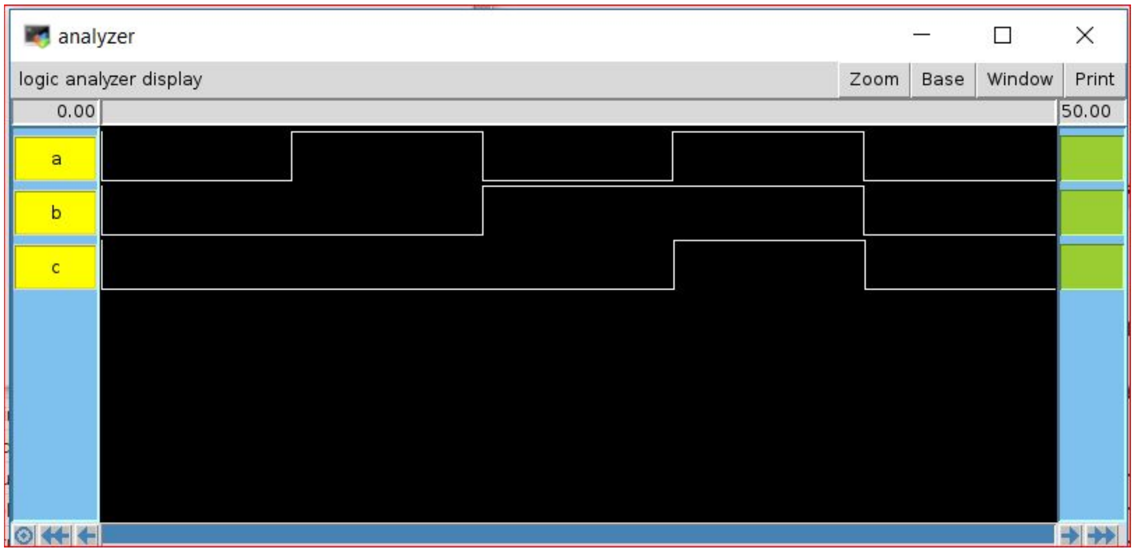
You can open the .cmd file with the "vi" editor to see the nodes for the analyzer and the vectors for inputs a and b. Now start irsim with the .sim file and the .cmd file. Be careful with the "-@" option that has a space before the .cmd file name.

```
irsim my_and.sim -@ my_and.cmd
```

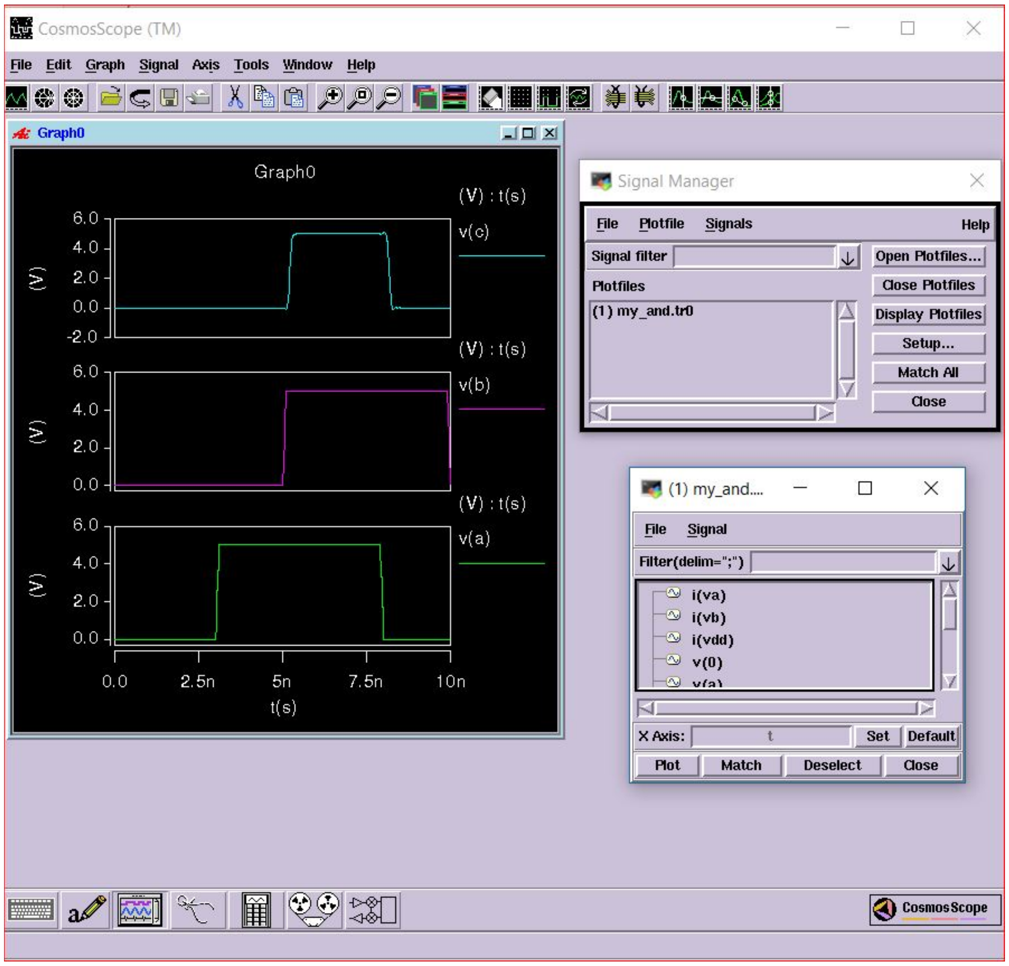
The analyzer graphics window should pop up with the simulation trace. Chose "`Zoom -> Full`" to see the entire simulation. You can use the "Snipping Tool" in Windows to save of screenshot of the Irsim window. You can also within Irsim "`Print -> File`" the results to a Postscript file with your choice of filename, for example `my_and_simulation.ps`. In my example, the Postscript file can be viewed with ghostscript or "evince" on Linux:

```
evince  my_and_simulation.ps
```

Hopefully the layout simulates correctly. You can quit irsim from the tkcon console window. You can then make new simulator .cmd files and start irsim again. If the logic and layout are correct, then we have completed the example design flow.



12. Simulation with hspice. We will describe the experiment file more closely in additional notes on Spice. A screen shot of the AND behavior is below.


CosmosScope (TM)
File Edit Graph Signal Axis Tools Window Help
Graph0
Graph0
(V) : t(s)
v(c)
v(b)
v(a)
6.0
4.0
2.0
0.0
-2.0
(V)
0.0
2.5n
5n
7.5n
10n
t(s)
Signal Manager
File Plotfile Signals Help
Signal filter
Open Plotfiles...
Plotfiles
Close Plotfiles
(1) my_and.tr0
Display Plotfiles
Setup...
Match All
Close
(1) my_and....
File Signal
Filter(delim=";")
i(va)
i(vb)
i(vdd)
v(0)
v(a)
X Axis:
t
Set
Default
Plot
Match
Deselect
Close
CosmosScope