ESE 370 Style Guide

Created by: Martin Deng, October 2017

General Design Principles

Don't put everything into a single schematic. Divide your circuit into components, and test each component separately before putting the whole circuit together.

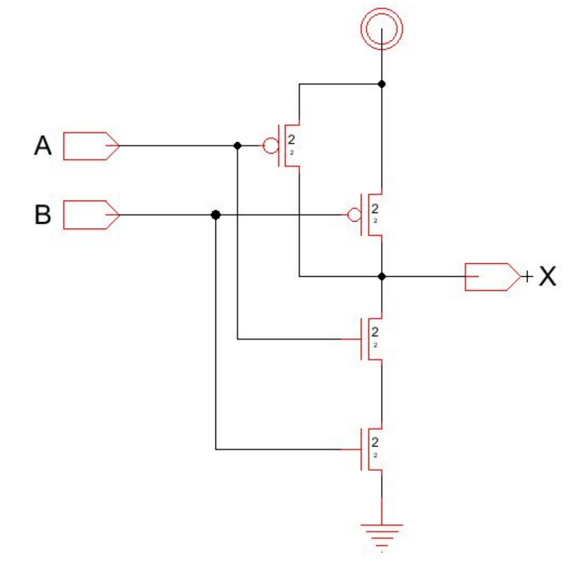
Have two kinds of schematics. Circuit designs and testbenches. Testbenches will help ensure that your circuit works the way you expect. (For those who have written software, think test-driven development)

Be comfortable with manually digging through SPICE files and checking that nodes are connected properly. Many errors are due to wires that look connected on screen but actually aren't, resulting in broken SPICE files.

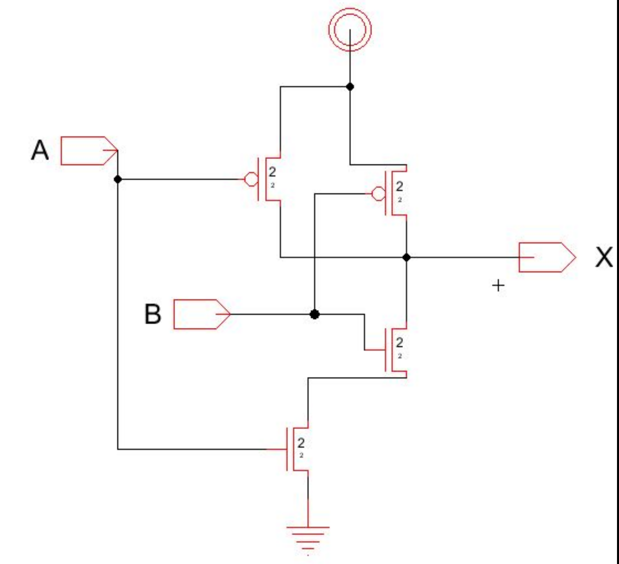
Circuit Designs

Keep your designs organized. Inputs on the left, outputs on the right, PMOS's in pull-up and NMOS's in pull-down networks, wires neat, and components lined up.

In every circuit make sure to include a global VDD and GND pin (double circles and stacked lines at top and bottom of diagrams below). All the global VDD pins will be connected together, same for the global GND pins. You will only need to put a single voltage source between a VDD and GND pin somewhere in your circuit or in a testbench circuit (see later) in your simulation. Measuring the current through this voltage supply source will allow you to later calculate the energy used by your circuit.



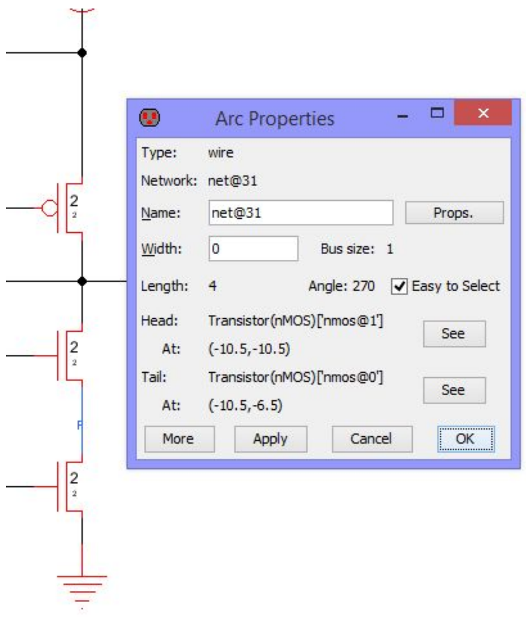
Good Example



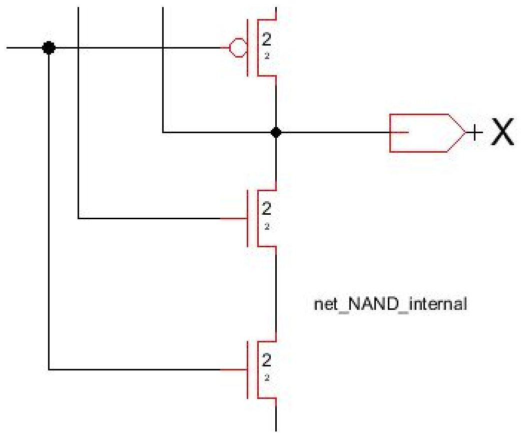
Poor Example

Electric will automatically label wires/nodes you export from your circuit and automatically generate names for all other wires/nodes. However, it is a good idea to manually label these so you can quickly identify them in the SPICE file and plot their voltages if needed.

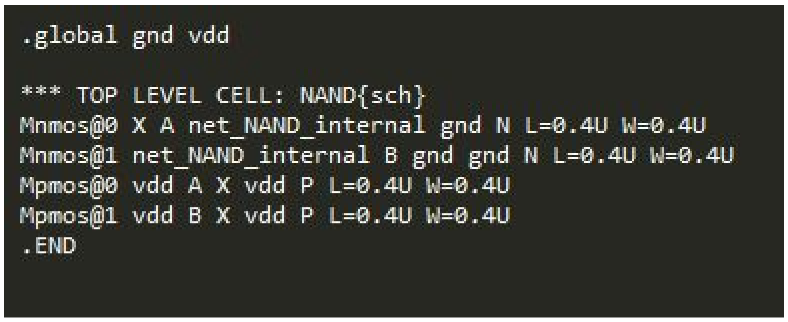
To do this, double-click on a wire. A properties window should appear.


Arc Properties

Type: wire
Network: net@31
Name: net@31
Width: 0 Bus size: 1
Length: 4 Angle: 270 [✓ Easy to Select]
Head: Transistor(nMOS)['nmos@1']
At: (-10.5,-10.5)
Tail: Transistor(nMOS)['nmos@0']
At: (-10.5,-6.5)



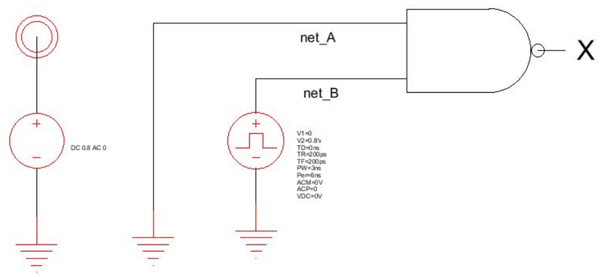
Electric automatically generates a name in the form “net@random_number”. A good naming convention is “net_usefulname”. Searching for “net@” in a SPICE file lets you quickly find automatically generated names which might indicate improperly created nodes if you have named all the ones you intended to create.

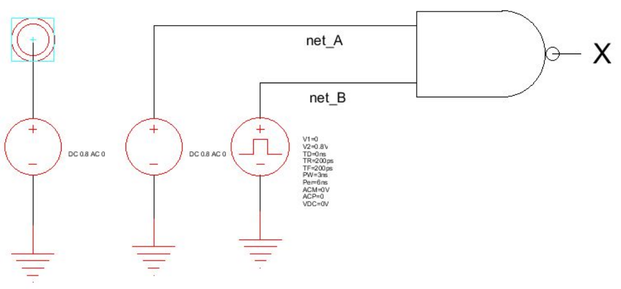


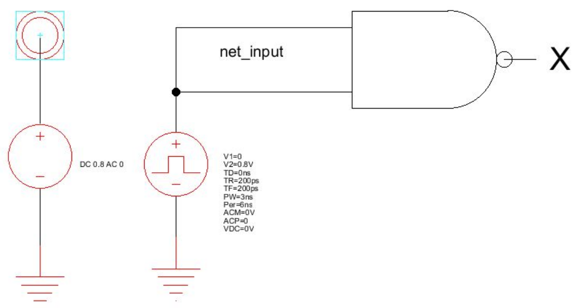
```
.global gnd vdd

*** TOP LEVEL CELL: NAND{sch}
Mnmos@0 X A net_NAND_internal gnd N L=0.4U W=0.4U
Mnmos@1 net_NAND_internal B gnd gnd N L=0.4U W=0.4U
Mpmos@0 vdd A X vdd P L=0.4U W=0.4U
Mpmos@1 vdd B X vdd P L=0.4U W=0.4U
.END
```

Testbenches

Generally, you will have many more testbenches than circuits in your project. Being meticulous and methodical in your testing will save many hours of debugging and plotting random voltages down the line. To test the NAND gate, we've created three separate testbenches.







Now we have a set of tests for our NAND gate that we can reuse even if the design of the NAND gate itself changes.

Remember to regenerate all your spice files whenever you make a change to a circuit or a testbench

Useful Ngspice commands

listing	Prints out currently loaded spice file

plot v(node@1) v(node@2) v(node@3)	Plot voltages at different locations in circuit
plot v(component@1.internal_node)	Plot voltage at node inside component@1
Meas is a very complicated command with syntax that takes several pages to cover. We will give a few examples of useful commands that can be modified for your work.	
meas tran tdelay trig node@1 val=0.1 rise=1 targ node@1 val=0.9 rise=1	Tdelay: Measures time it takes for voltage at node@1 to go from 0.1V to 0.9V (rise time) Trig: trigger/start measuring when node@1 is 0.1V Rise=1: first time voltage at node@1 rises to the trigger Targ: Target voltage of 0.9V to stop measurement
meas tran tdelay trig node@in val=0.5 rise=1 targ node@out val=0.5 fall=1	Measure time between voltage at node@in rising to 0.5 and voltage at node@out falling to 0.5 (propagation delay)
meas tran avgvar AVG node@out from=0ns to=10ns	Measure the average voltage of node@out from 0ns to 10ns and put in the variable avgvar Replace AVG with the following if needed RMS: root mean squared MIN: min MAX: max PP: peak to peak
meas tran curvar FIND i(vsupply@0) WHEN v(node@in)='0.25*1V'	Find the current through voltage supply vsupply@0 when the input voltage reaches 0.25V
meas tran volvar FIND node@out AT=1ns	Find the voltage at node@out when time is 1ns
meas tran integvar INTEG i(vsupply@0) from=0ns to=10ns	Integrate the current from vsupply@0 from 0ns to 10ns and store in integvar

The full Ngspice user manual can be found below:

http://ngspice.sourceforge.net/docs/ngspice25-manual.pdf