EE194/EE290C Lab 4
Analog, digital, mixed-signal setup instructions
Spring 2018

Homework:

1. Characterize NMOS & PMOS of our 32nm class process. This is to get familiar with the tool chain; we will redo this characterization with the real ST PDK when we've received access to it.

This must include:
Id vs VDS curves with changing VGS for W/L=100nm/30nm. Both VDS and VGS should sweep from 0 to 1V (or -1 to 0V for PMOS).

All below should be characterized with W/L=100nm/30nm and W/L=1um/0.3um, VDS=0.5V (or -0.5V for PMOS), and sweeping VGS from 0 to 1V (or -1 to 0V for PMOS)
Id -- Do they look like short or long-channel devices? If short, estimate Vt; if long, replot sqrt(Id) and estimate Vt.
gm
ro -- what's a value of ro you can depend on over a 0.5V output swing?
gm*ro -- what's the best intrinsic gain you can get in this process?
ft (gm/Cgs) -- what's the highest speed we can ever get from these transistors?

2. Analog: Simulate VOUT vs VIN of an analog inverter of minimum size and ensure it works.

3. Digital: Simulate a 4-bit counter (code below) with resistive load.

4. This is where we'd ask you to simulate the counter with the inverter, but right now the digital simulator, AMS, can't interpret our class models. We're trying to figure out a workaround.

counter.v:

```
//-----------------------------------------------------
// Design Name : counter
// File Name   : counter.v
// Function    : 4 bit up counter
// Coder       : Deepak, asic-world.com
//-----------------------------------------------------
module counter (clk, reset, enable, count);
input clk, reset, enable;
output [3:0] count;
reg [3:0] count;
```

```
always @ (posedge clk)
if (reset == 1'b1) begin
  count <= 4'b0000;
end else if ( enable == 1'b1) begin
  count <= count + 1'b1;
end

endmodule
```

ANALOG SIMULATION

Use NoMachine, x2go, SSH with an X server, etc., to connect to an hpse machine, e.g., hpse-[9...15].eecs.berkeley.edu and start a new session. You can also try a c125m-[1...24].eecs.berkeley.edu machine. If you don't already have an account, you should be able to get one by going to https://inst.eecs.berkeley.edu/webacct/. Depending on your luck, this might be the most difficult step of the homework!

Edit your ~/.bashrc and ~/.bash_profile to source the class config file by adding this line to the end:

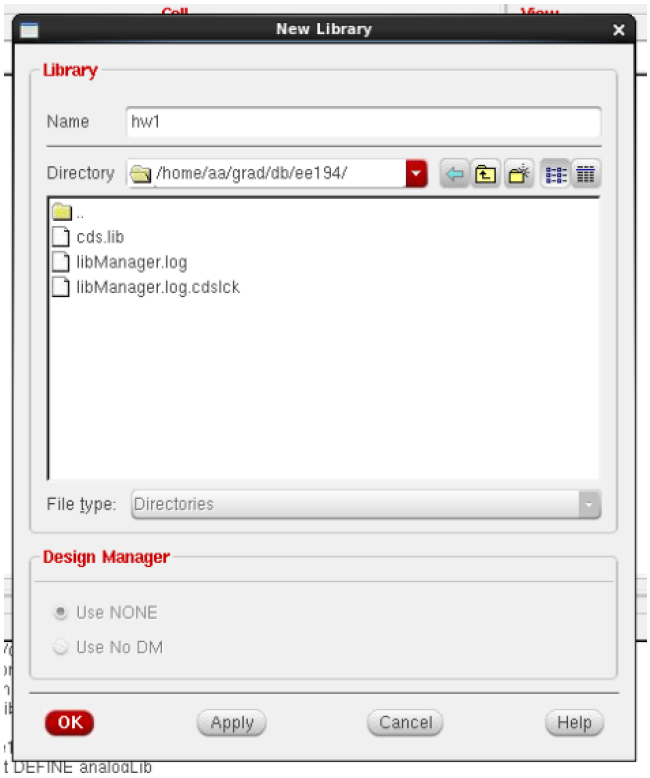
```
source /home/aa/grad/db/ee194-shared/ee194.bashrc
```

Log out and log back in to enable these changes.

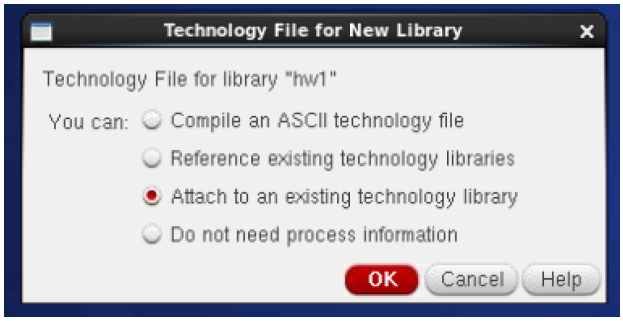
Create a new virtuoso working directory, copy the default cds.lib, link .cdsinit, and launch virtuoso inside it (NOT virtuoso6 as in other classes)

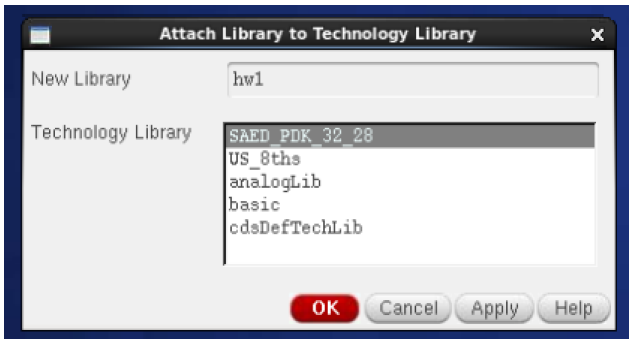
```
cd ~
mkdir ee194
cd ee194
cp ~db/ee194-shared/cds.lib .
cp ~db/ee194-shared/.cdsinit .
virtuoso&
```

Open Library Manager and create a new library:

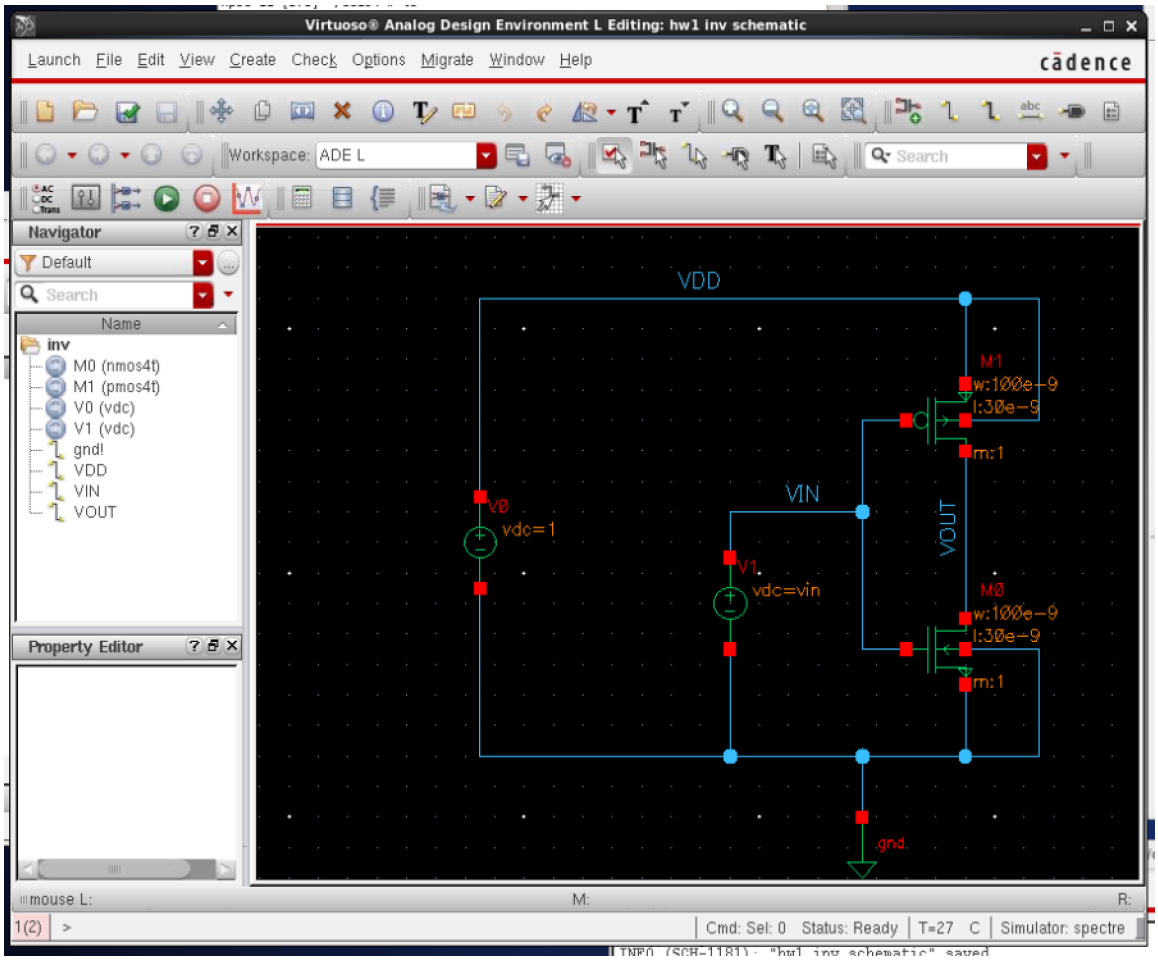


Attach to an existing library:

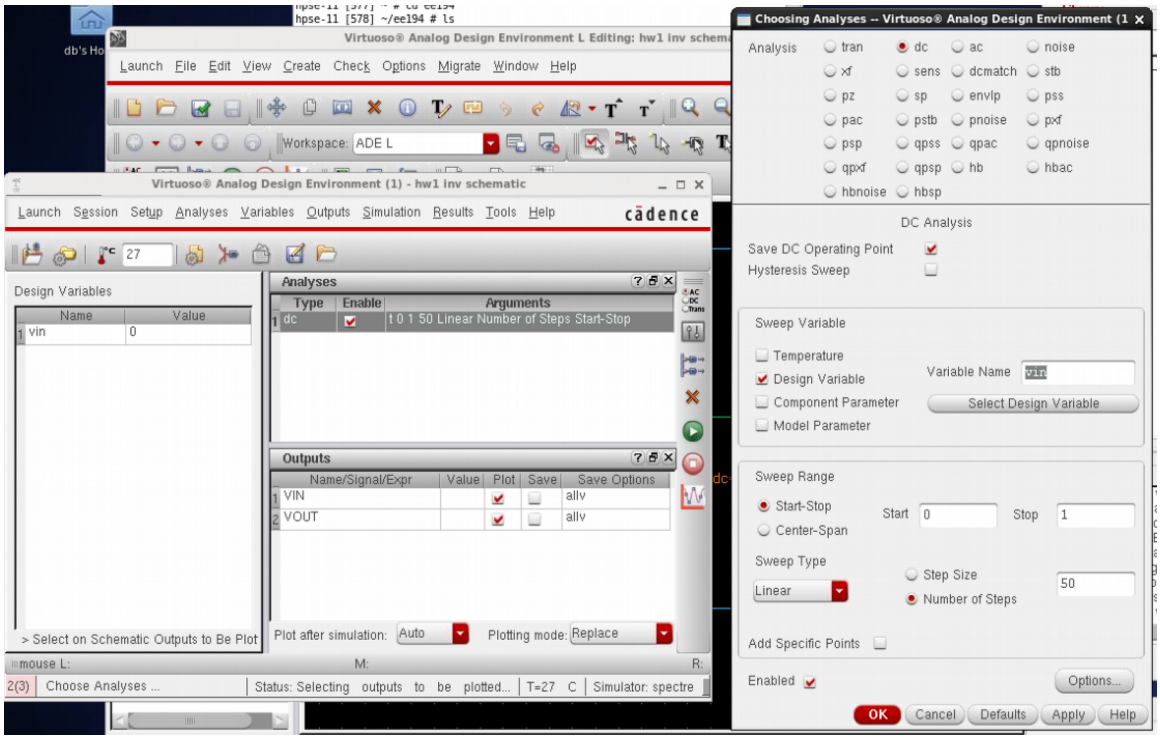




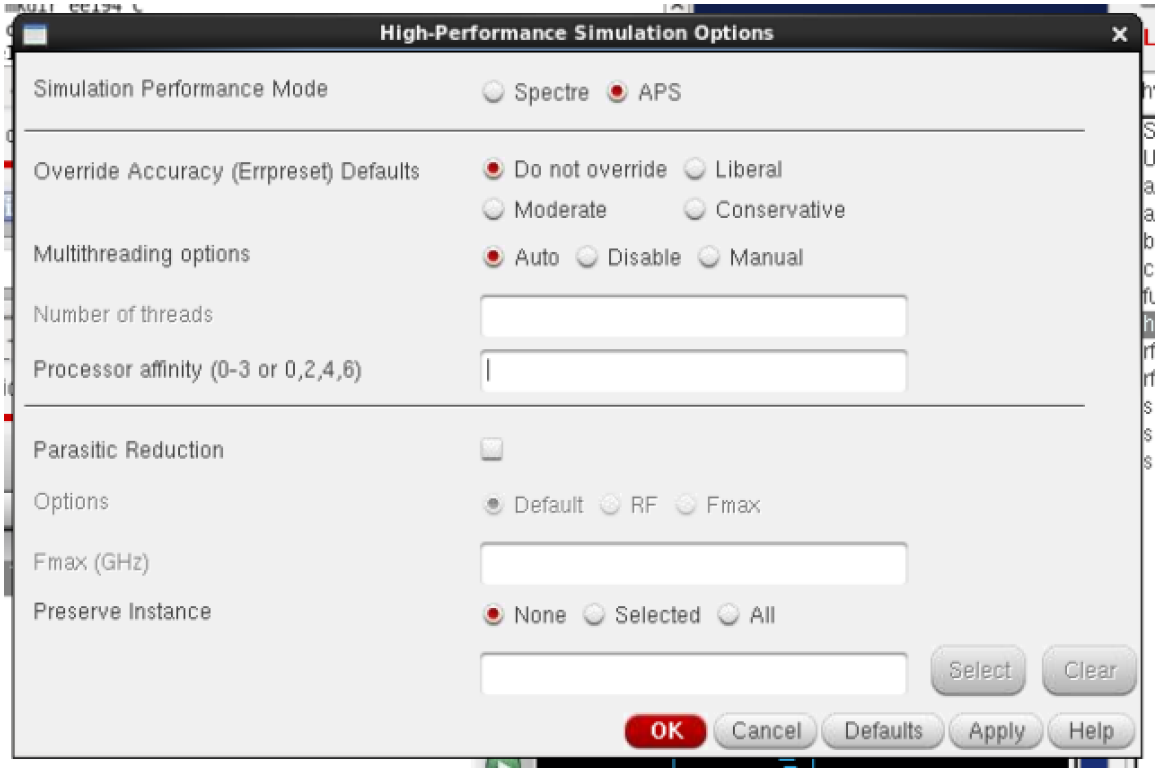
Create a new schematic and add an nmos4t and pmos4t with default sizes from the SAED_PDK_32_28 library, and a VDD (set to 1V) and VIN (set to variable VIN) DC voltage source and gnd from analogLib, then check and save:



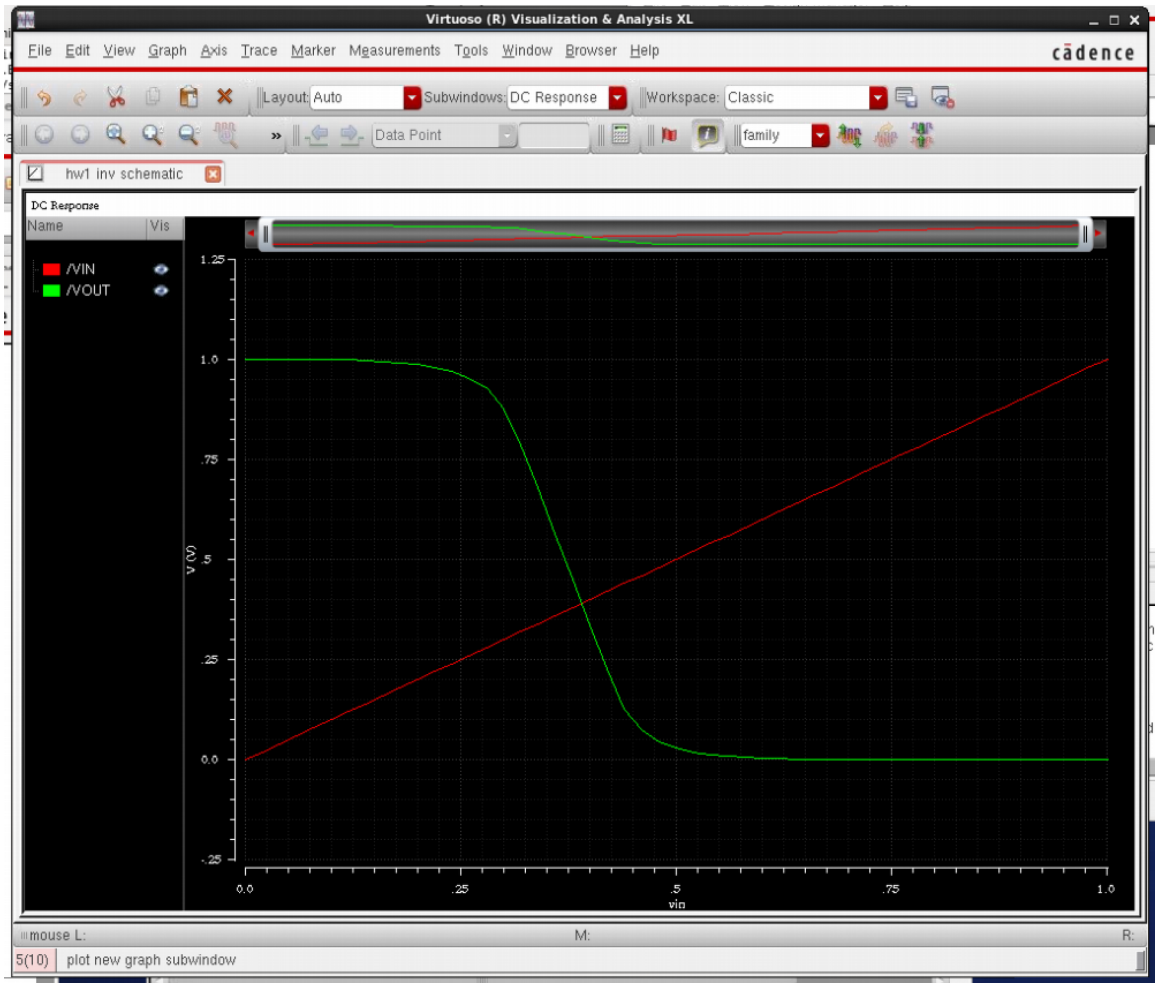
Launch ADE L and set up a DC sweep from 0 to 1V, with VIN and VOUT plotted:



In ADE L, you can go Setup > High Performance Simulation and change from Spectre to APS. This will speed up your simulations considerably. For this test the speedup won't matter but it's a good idea to get in the habit of running with APS to save time.

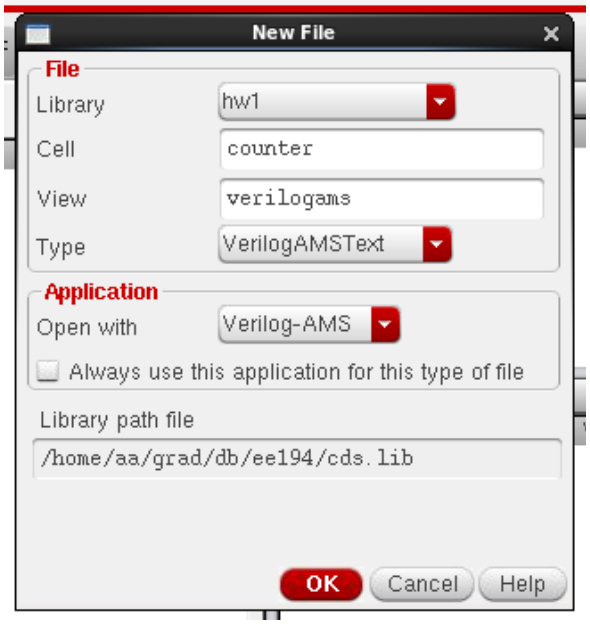


Run the simulation and ensure the results look inverter-ish, shown below. If the output is a straight line, try simulating with HSPICE instead. Switch inside ADE L by going Setup > Simulation/Directory/Host and selecting HSPICE from the first drop-down menu. I've noticed that switching to HSPICE works, and afterward switching back to spectre also works. Once your simulation is in a working state, I strongly suggest saving that simulation state by going Session > Save State and saving as a cell view.

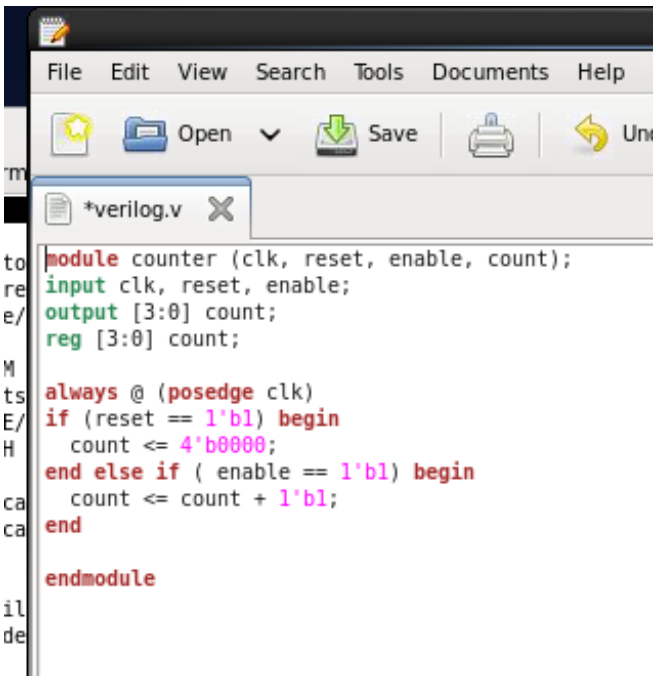

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
Layout: Auto
Subwindows: DC Response
Workspace: Classic
Data Point
family
hw1 inv schematic
DC Response
Name
Vis
/VIN
/VOUT
1.25
1.0
.75
.5
.25
0.0
-.25
V
0.0
.25
.5
.75
1.0
vin
mouse L:
M:
R:
5(10)
plot new graph subwindow

MIXED-SIGNAL SIMULATION

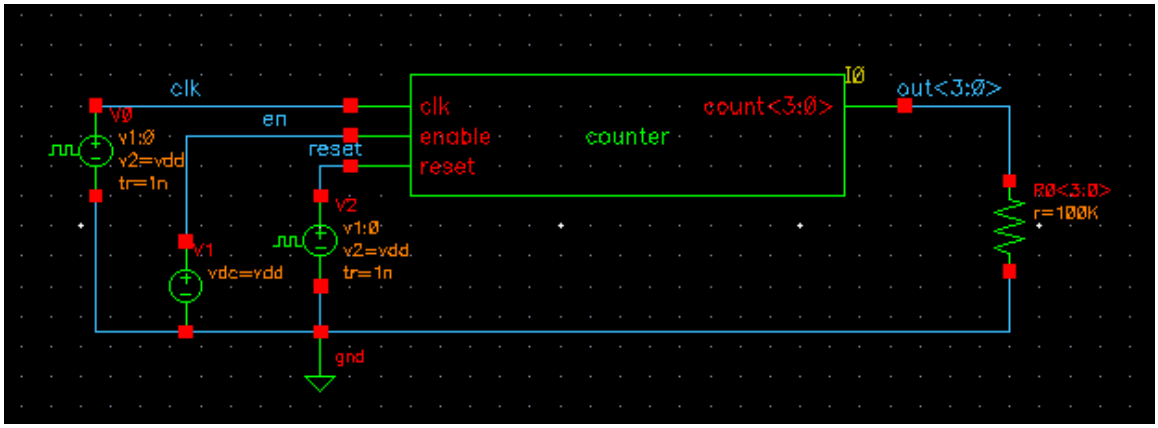
Create a new cellview in your existing hw1 library of type verilog-AMS:



You'll get a skeleton gedit window. Delete everything and paste in the code from the beginning of this document. The header causes some parsing issues so I've ignored it below. You can ignore the warnings popup after you save and close.

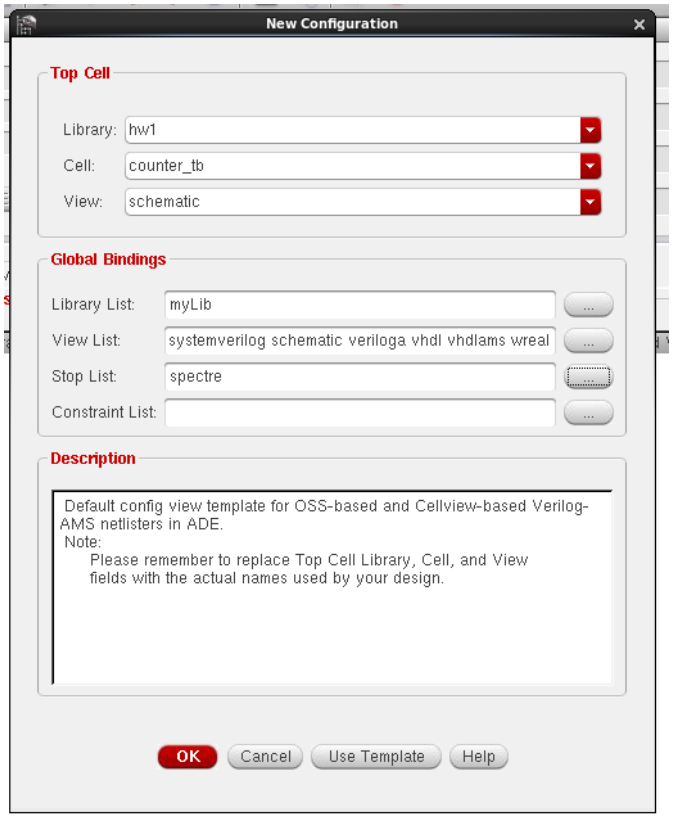


You should get a popup asking to generate a symbol. Say yes, then close the symbol. Create a new counter_tb and add the symbol along with signals and resistors to run and load the counter with 1V VDD:

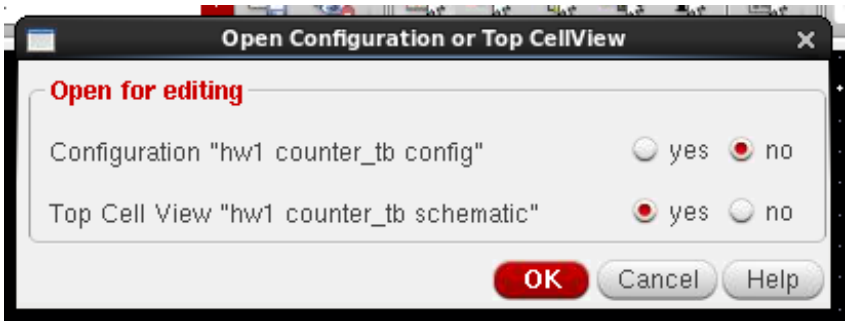


To simulate mixed-signal, we have to use a config file to wrap around our schematic. Create a new cell view next to your counter testbench schematic of type config:

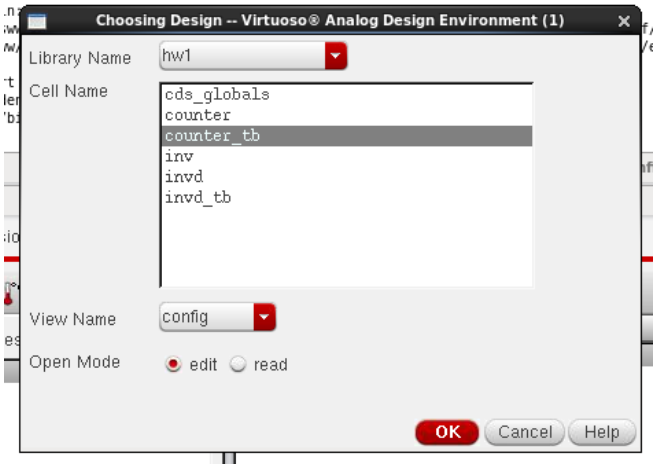
On the next screen, click “Use Template” and select AMS. Then change View at the top to schematic. The result should look like the screenshot below:



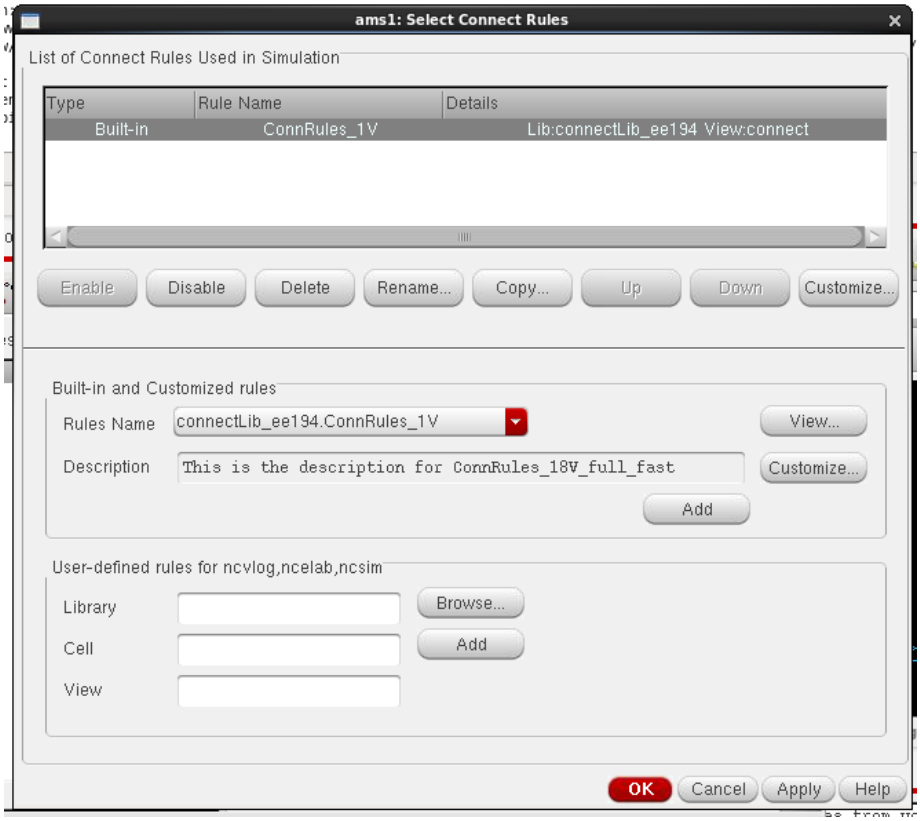
Click OK, save the config, and close the config window. Next, double-click on the config and select radio buttons to open schematic and not the config:



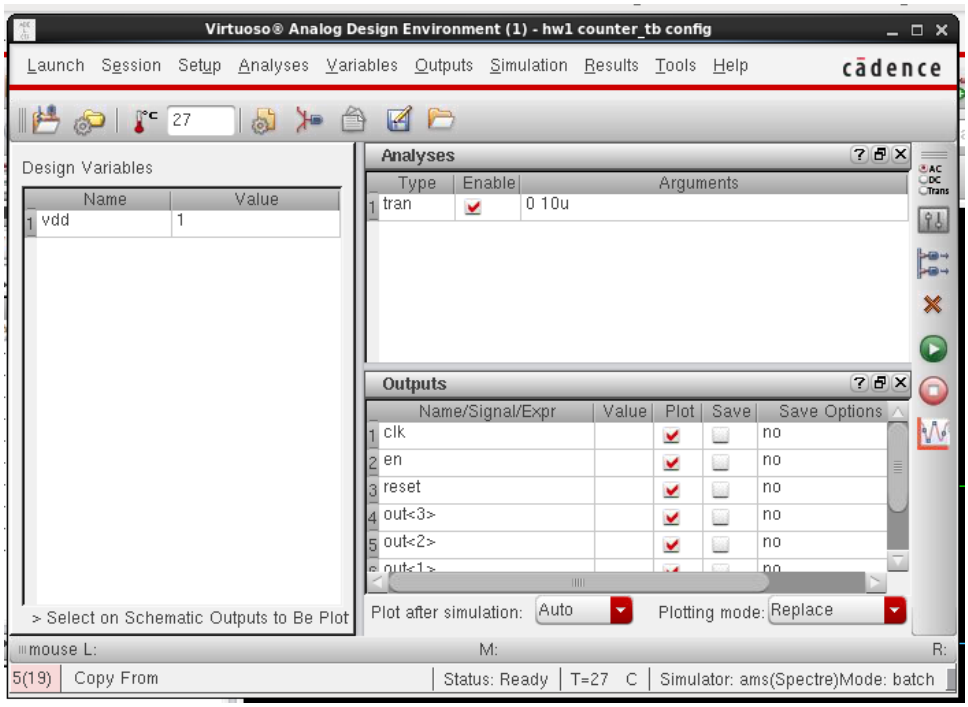
Launch ADE L, then go Setup > Design and verify the view name is config, NOT schematic:



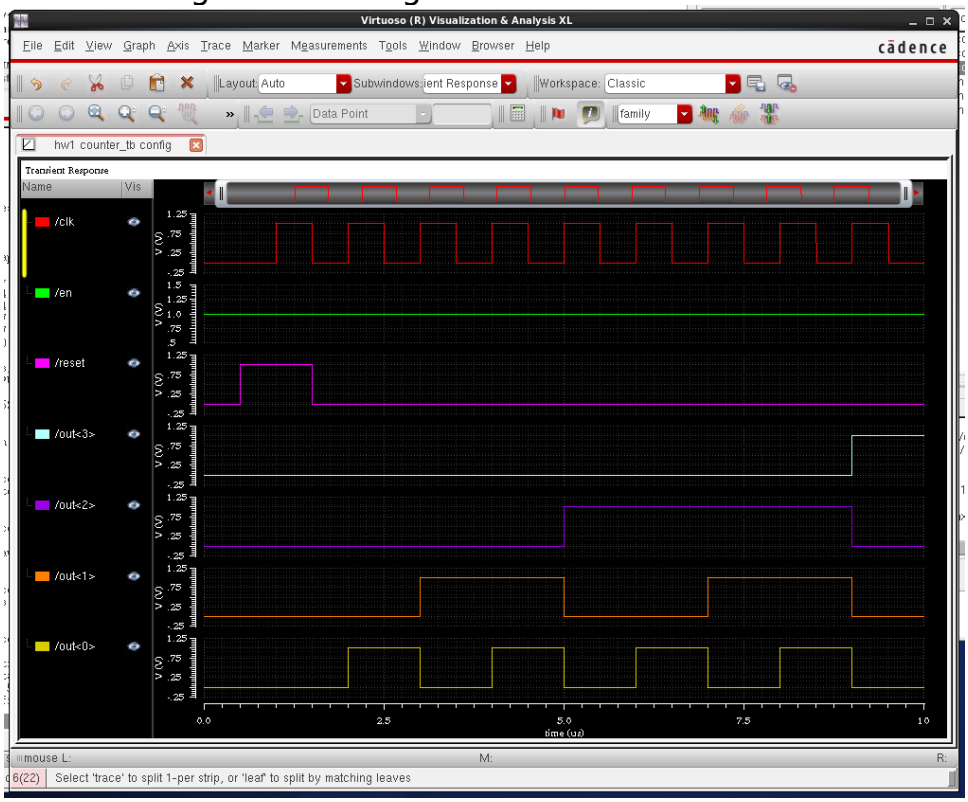
Then go Setup > Simulator/Directory/Host and change the simulator to “ams” and click OK. Next go Setup > Connect Rules. This is the screen where we define what counts as a digital 1 and 0 as input voltage, and what output voltage the verilog block will assert when it presents a 1 or 0 at its output. We have some connect rules for 1V VDD already set up for the class. By default, a 1.8V rule will be in this screen; delete it from the top third of the screen and add connectLib_ee194.ConnRules_1V instead. (Note the description will probably mention 1.8V.) The result should look like below:



Set up a transient simulation to ensure the counter is working:



You should get something that looks like a counter:



The end for now!