

**United** International **University** (UIU)

Dept. of Electrical and Electronic Engineering (EEE)

# **Course**: VLSI Design Lab (EEE 442)

**Experiment 1: Simulation of Multiplexers using Verilog**

Introduction:

1. Collect the login ID and password from the lab instructor, log into your account and familiarize yourself with the Linux Operating system. Practice the Linux commands provided in the attached document.
2. Copy the cshrc file to your home directory as .cshrc file

* cd -- moves you to your home directory
* cp ~sahmed/cshrc .cshrc

1. Switch to csh, this will setup your cadence tools environment variables

* csh

1. Create a lab directory and a simulation sub-directory under your work area:

* mkdir lab1
* cd lab1
* mkdir simulation
* cd simulation

1. Use an editor (vi, nedit, gedit etc.) to write the Verilog code for a 2-to-1 multiplexer (MUX)

* vi mux21.v

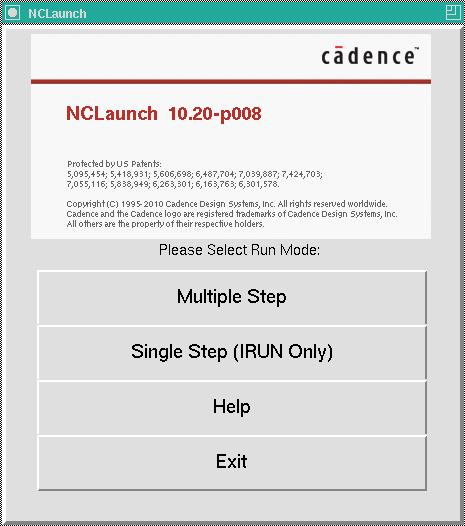
1. Use an editor to write a test bench Verilog code for your 2-to-1 MUX

* vi mux21\_test.v

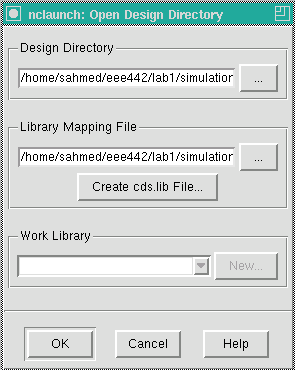
1. We will use Incisive Enterprise Simulator (IES) to verify that the MUX is working properly. Invoke the tool by typing ‘nclaunch –new’ in the terminal

* nclaunch –new

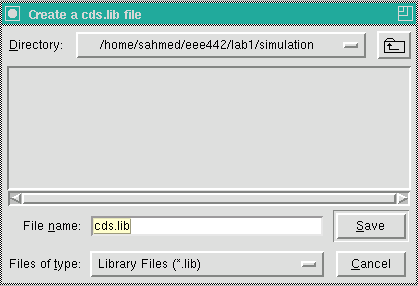
1. NCLaunch window will appear. Click on the ‘Multiple Step’



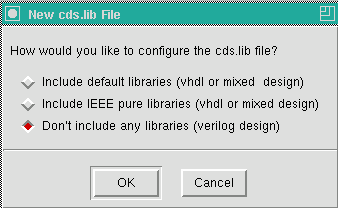
1. ‘nclaunch: Open Design Directory’ window will appear. Click on the ‘Create cds.lib File’



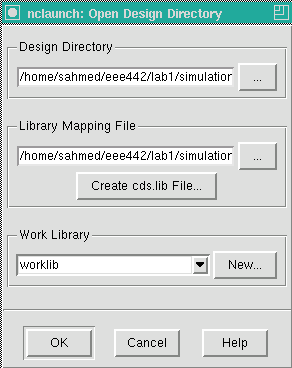
1. ‘create a cds.lib file’ window will appear. Click on ‘Save’



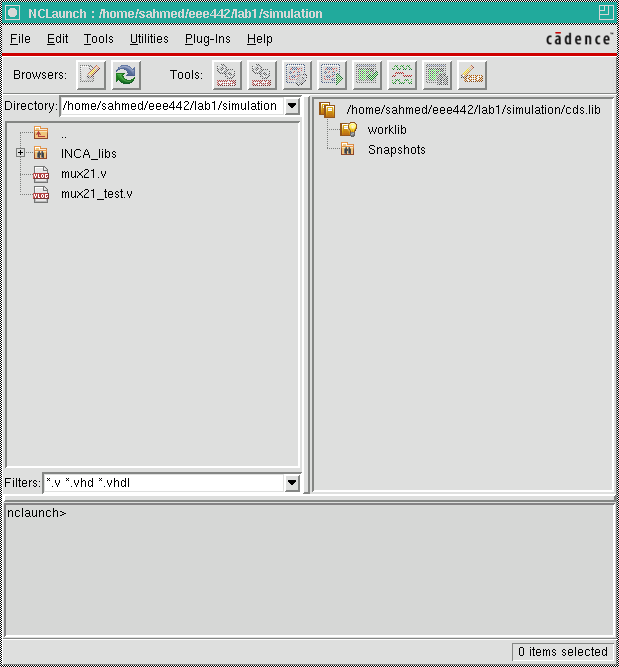
1. ‘New cds.lib File’ window will appear. Select the third option since we are using Verilog. Click on ‘OK’



1. Click ‘OK’ in the ‘nclaunch: Open Design Directory’ window.

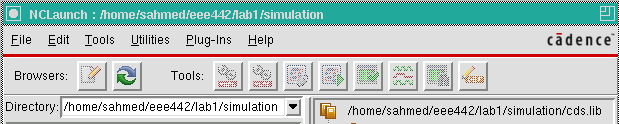


1. In the NCLaunch window we will see the design (mux21.v) and the testbench (mux21\_tb.v) that we created in the *simulation* directory.



Command Entry Window

1. Now we will COMPILE the code to check their syntax and semantics. Select both the design (mux21.v) and the testbench (mux21\_test.v) and click on the *ncvlog* for Verilog designers.



1. Any error in the code will be reported in the ‘Command Entry Window’ (at the bottom of the NCLaunch). Clean all the errors in the design and/or the testbench.

ELABORATION – constructs design hierarchy and connects signals

1. Open the ‘worklib’ directory on the right side of the window and select the test bench as it is the top module (which contains the design module). Click on the ‘launch elaborator (*ncelab*)’

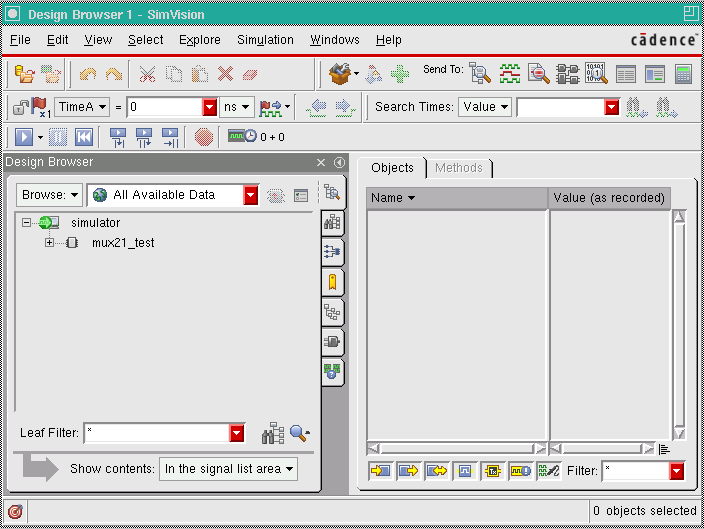


SIMULATION – executes simulation code

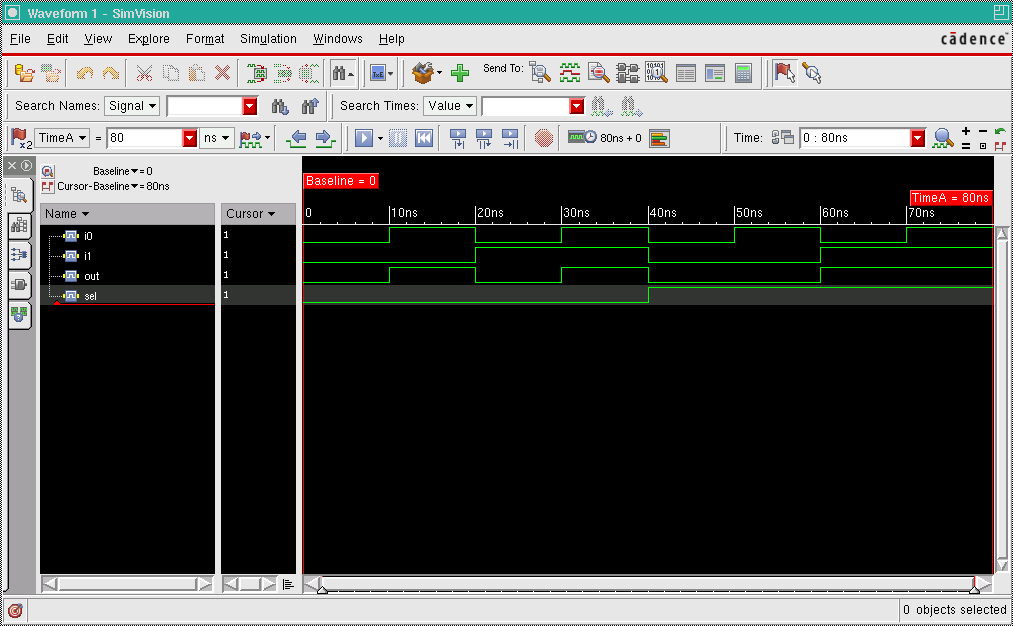
1. Open the ‘snapshots’ directory on the right side of the window and select the worklib of the test bench. Click on ‘launch simulator’



1. ‘Console – SimVision’ window will appear. This is used to perform simulation in command mode. Minimize this window since we will be using GUI mode for simulation.
2. ‘Design Browser – SimVision’ window also appears. There are many options in the ‘Design Brwoser’ to analyze the design and debugging. Select the test bench module on the left side of the window. Click on the ‘waveform’



1. ‘Wavefor – SimVision’ window will appear. In the waveform window we will see the ports of the design. Click on the ‘run simulation’ to start the simulation. Use the ‘pause’ button to interrupt or to stop simulation. Use different options like zoom in, zoom out etc. to analyze the plot. There are many options in the ‘Waveform’ window to analyze the design and debugging.



run simulation

1. After verifying the design, close the tools.
2. Write Verilog code for a 4-to-1 MUX (mux41.v) using the 2-to-1 MUX you have just tested.
3. Write the test bench (Stimulus Module) for the above 4-to-1 MUX (mux41\_test.v). The test bench code must be according to the following information:

MUX Inputs: i0, i1, i2, i3

Selector inputs: s0, s1

The outputs: out = i0; for s0 = 0 and s1 = 0

out = i1; for s0 = 1 and s1 = 0

out = i2; for s0 = 0 and s1 = 1

out = i3; for s0 = 1 and s1 = 1

Choose the input values (clocks) as given below:

i0 = > Low = 05, High = 05

i1 = > Low = 10, High = 10

i2 = > Low = 20, High = 20

i3 = > Low = 40, High = 40

s0 = > Low = 80, High = 80

s1 = > Low = 160, High = 160

1. Use Incisive Enterprise Simulator (IES) to verify that the MUX is working properly. Invoke the tool by typing ‘nclaunch’ without the ‘– new’ option since we have will use the previously created worklib.

* nclaunch

1. Follow the same steps described before to test the 4-to-1 MUX.
2. Write behavioral model of an 8-to-1 MUX. Must utilize vector notations for inputs/selectors. Provide appropriate stimulus to run simulation.
3. Use Incisive Enterprise Simulator (IES) to verify that the 8-to-1 MUX is working properly. Follow the same steps described before.

