Xcelium Tutorial

NORTHWESTERN
UNIVERSITY

McCormick

Northwestern Engineering

Xcelium Tutorial

Before going to next steps, please note that those lines that start with '#' are explanation, lines that follow with '$' are commands and you need to type **in your terminal and press enter.**

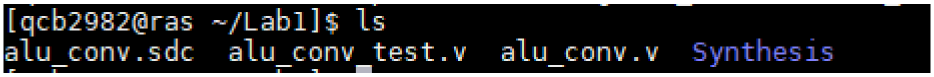
1. RTL simulation

In this part, you only need the verilog code (RTL) "alu_conv.v" and its testbench "alu_conv_test.v".

1) # Go into the directory "Lab1" and copy alu_conv_test.v to this folder.

```
        $ cd ./Lab1
$ cp /vol/ece303/genus_tutorial/alu_conv_test.v .
```

Lab1 folder should contain: alu_conv.v , alu_conv_test.v. You could type "ls" to see files in the directory.



```
[qcb2982@ras ~/Lab1]$ ls
alu_conv.sdc  alu_conv_test.v  alu_conv.v  Synthesis
```

(If you have not done Genus_Tutorial, you can simply run the following command to copy the files so you can still follow this tutorial to learn Xcelium.

```
        $ mkdir Lab1
        $ cd ./Lab1
        $ cp /vol/ece303/genus_tutorial/alu_conv.v .
```

)

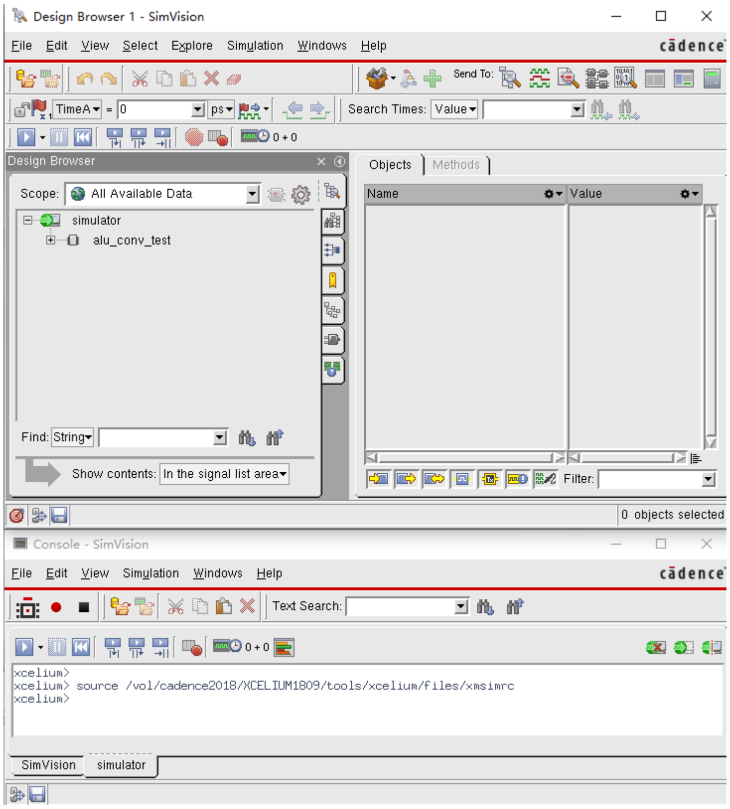
2) # Enter the following commands to source the cadence environment

```
        $ source /vol/ece303/genus_tutorial/cadence.env
```

```
[qcb2982@ras ~/Lab1]$ source /vol/ece303/genus_tutorial/cadence.env
```

3) # Type the following command to run the cadence tool. In this case you will see a new open window like this:

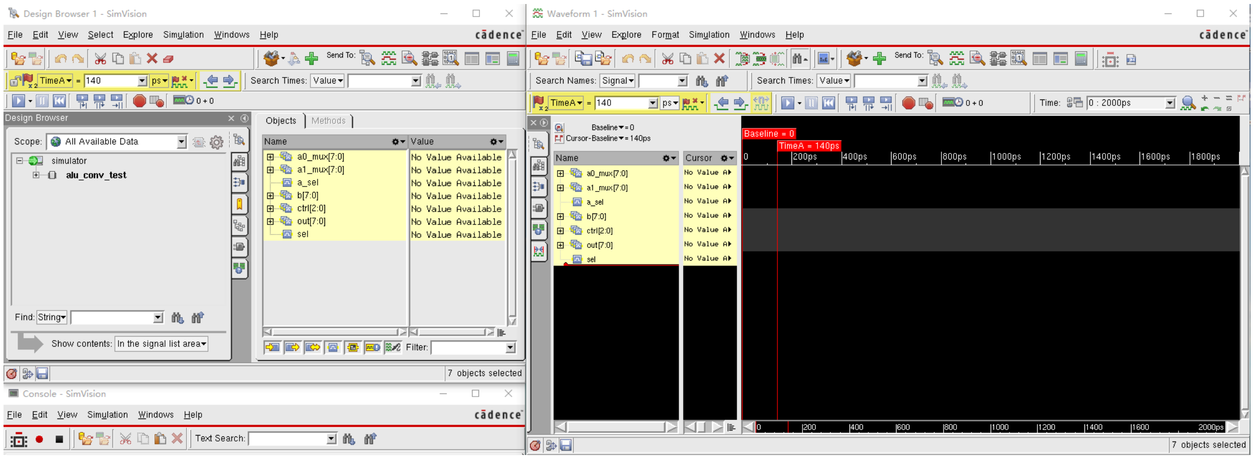
```
        $ xrun -64bit -gui -access r alu_conv.v alu_conv_test.v
```



```
xcelium>
xcelium> source /vol/cadence2018/XCELIUM1809/tools/xcelium/files/xmsimrc
xcelium>
```

Important: the terminal should show no error, or the new window will not pop-up.

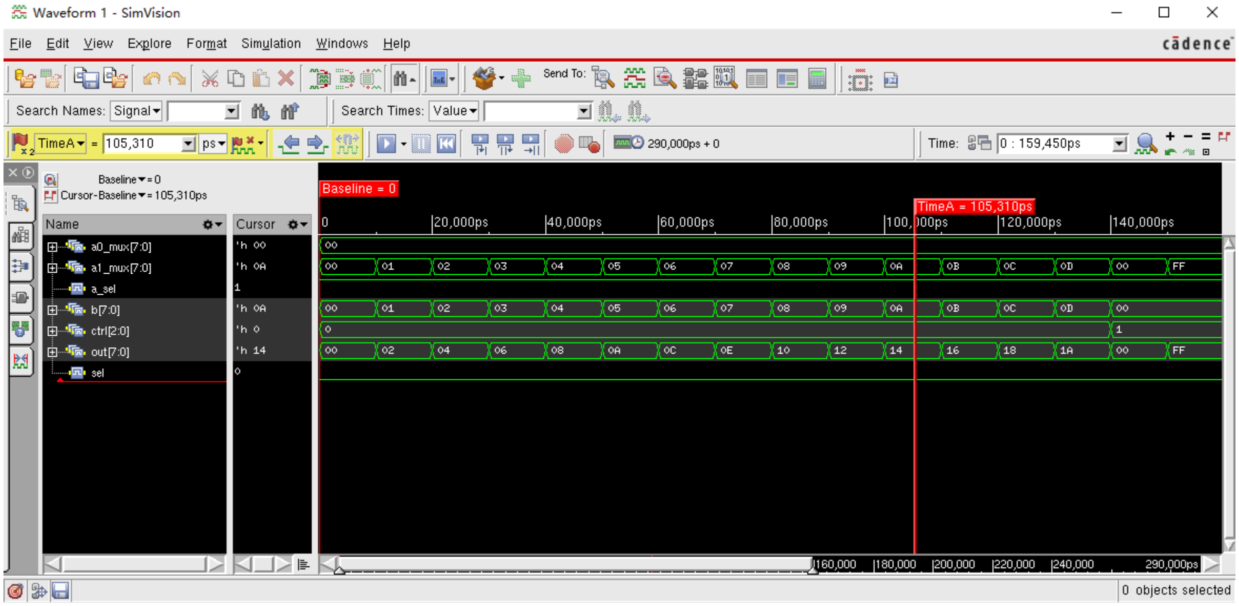
4) # In this window you can select the signal you want to send to the **Waveform Window** (right click a signal -> Send to Waveform Window)



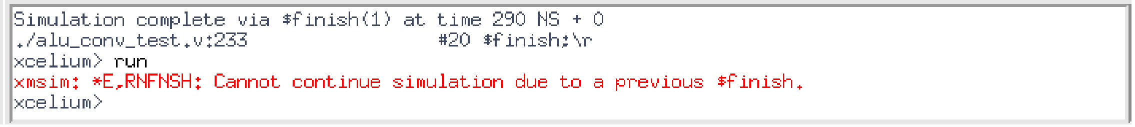
Name	Value
a0_mux[7:0]	No Value Available
a1_mux[7:0]	No Value Available
a_sel	No Value Available
b[7:0]	No Value Available
ctrl[2:0]	No Value Available
out[7:0]	No Value Available
sel	No Value Available

5) # In the Waveform SimVison window you can simulate your design. Simulation->Run.

Note: You need to use zoom in/out function to pick the expected observation time period.



Important: if you see error bellow, click "Reset the simulation back to time 0". Then run simulation again.

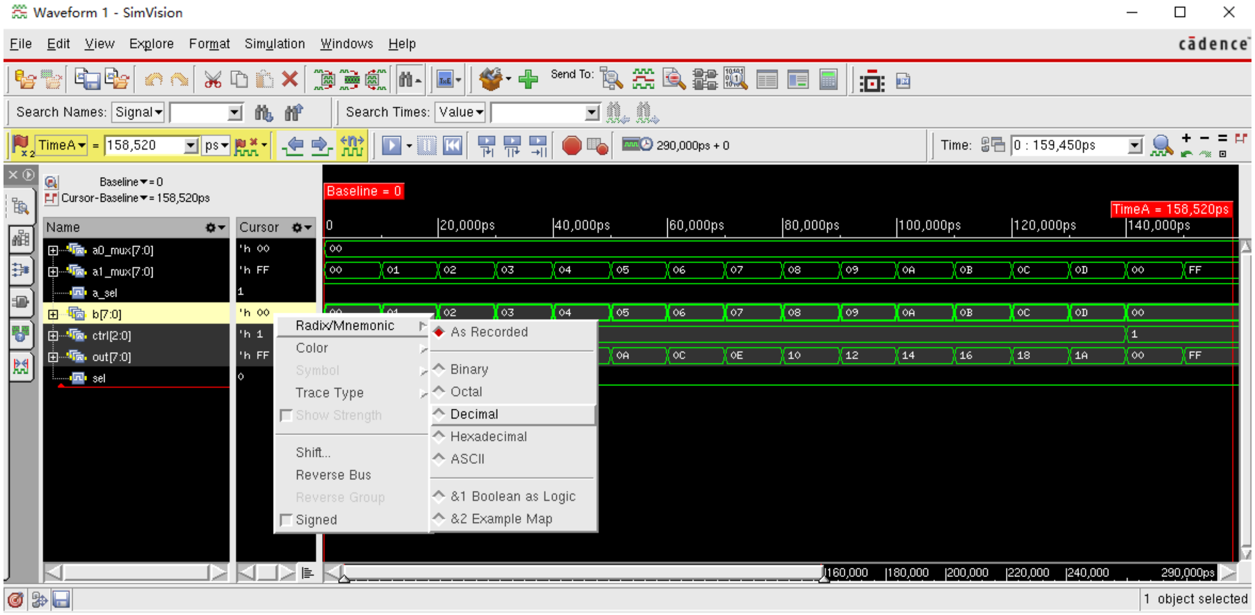


```
Simulation complete via $finish(1) at time 290 NS + 0
./alu_conv_test.v:233                    #20 $finish;\r
xcelium> run
xmsim: *E,RNFNSH: Cannot continue simulation due to a previous $finish.
xcelium>
```

6) # Verify the result of RTL code

Based on the observation, you can confirm whether the function is correct of your RTL code.

Right click column of Cursor to change value from Hex to Decimal or Binary, so that you can check whether the result is correct.



Note: Close all windows when you finish this session.

2. Gate level netlist simulation:

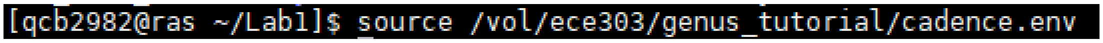
To do this part, you need to first finish the Genus lab. Check the other tutorial "Genus Tutorial". After you finish the Genus lab, you should have a new verilog file: alu_conv_syn.v under "Synthesis" folder. (This is the gate level netlist file)

1) # Go into the directory "Lab1"

   $ cd ./Lab1

2) # Enter the following commands to source the cadence environment

   $ source /vol/ece303/genus_tutorial/cadence.env



```
[qcb2982@ras ~/Lab1]$ source /vol/ece303/genus_tutorial/cadence.env
```

3) # Type the following command to run the cadence tool.

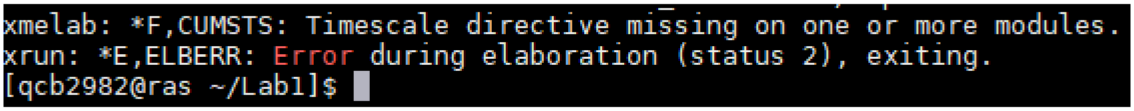
   $ xrun -64bit -gui -access r -xmelab_args "-warnmax 0 -delay_mode zero -maxdelays" ./Synthesis/alu_conv_syn.v alu_conv_test.v /vol/ece303/genus_tutorial/NangateOpenCellLibrary.v

Important: the commands inside "" are options for "xmelab" which is an elaboration function called for gate level netlist simulation. alu_conv_syn.v is the gate level netlist which was generated in genus tutorial.

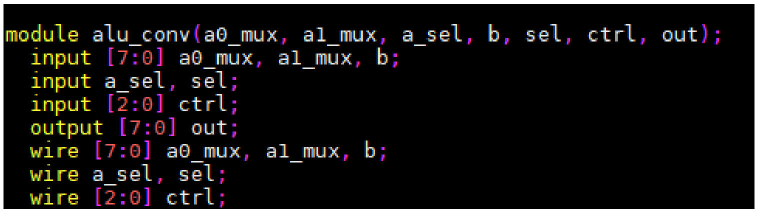
Note: If you see error below, add one line

`timescale 1ns/10ps

at the top of alu_conv_syn.v file and enter command again. Generally, we need the above line for each Verilog file to be simulated.

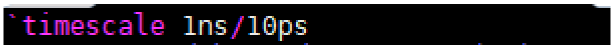


```
xmelab: *F,CUMSTS: Timescale directive missing on one or more modules.
xrun: *E,ELBERR: Error during elaboration (status 2), exiting.
[qcb2982@ras ~/Lab1]$
```



```
module alu_conv(a0_mux, a1_mux, a_sel, b, sel, ctrl, out);
  input [7:0] a0_mux, a1_mux, b;
  input a_sel, sel;
  input [2:0] ctrl;
  output [7:0] out;
  wire [7:0] a0_mux, a1_mux, b;
  wire a_sel, sel;
  wire [2:0] ctrl;
```

Add this line at the top of the file

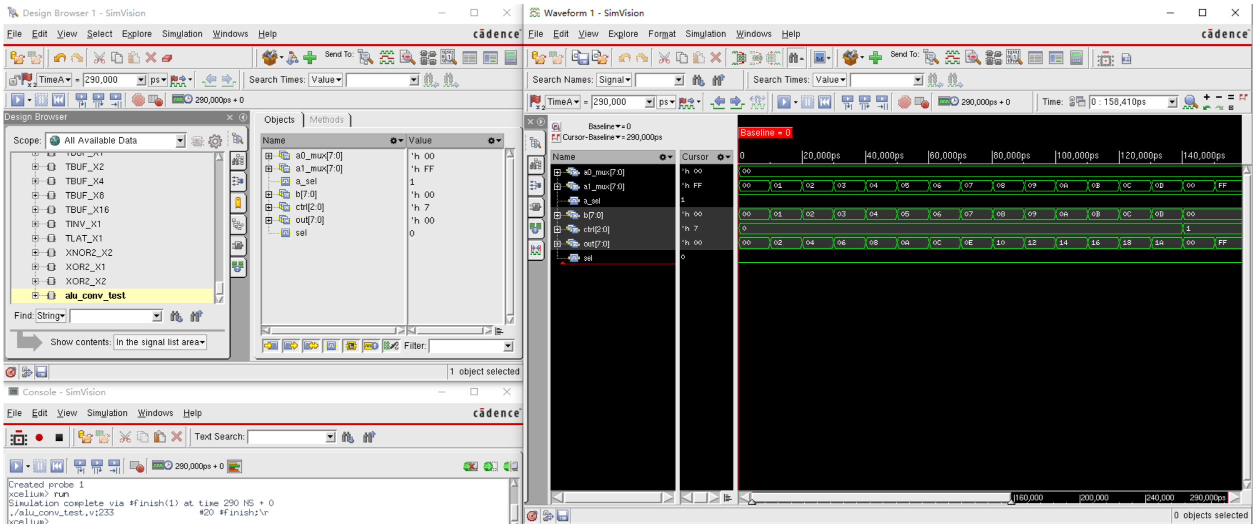


```
`timescale 1ns/10ps
```

After modification, save change and quit, then go back to setp 3 to type the same command.

4) After last command, there will be a **SimVison** window pop-up, in this window you can select the signal you want to send to the **Waveform Window** (right click a signal -> Send to Waveform Window). Scroll down to the bottom and send signals in alu_conv_test to waveform window.

   In the Waveform SimVison window you can simulate your design. Simulation->Run.



Name	Value
a0_mux[7:0]	'h 00
a1_mux[7:0]	'h FF
a_sel	1
b[7:0]	'h 00
ctrl[2:0]	'h 7
out[7:0]	'h 00
sel	0

```
Created probe 1
xcelium> run
Simulation complete via $finish(1) at time 290 NS + 0
./alu_conv_test.v:233                    #20 $finish;\r
xcelium>
```

The result of the gate level simulations should match with the RTL simulations. But simulation is based on gate level netlist this time.

Note in this tutorial, we did not give timing information to each gate, which means each gate has zero delay. We can perform simulation with real delay of gate, which is called "back-annotated" simulation. This operation needs to modify the xrun command. We skip this operation in this tutorial.