

Tutorial for Verilog Synthesis Lab (Part 1)

In this lab, you will be required to write a verilog code for serial signed-numbers multiplier, then simulate and synthesize it. Later you will do place and route, then tape out. It is assumed that you have very good knowledge on verilog as prerequisite for this lab. You are required to write your own verilog code for the lab itself. The sample Verilog code discussed in this tutorial is not the actual code required to complete the lab. The sample code is to familiarize yourself with tools you will be required to use to complete the lab.

Synthesis is the process of converting a high-level description of design (Verilog/VHDL) into an optimized gate-level representation. Logic synthesis uses a standard cell library which have simple cells, such as basic logic gates like AND, OR, and NOR, or macro cells, such as adder, muxes, memory, and flip-flops. Standard cells put together are called technology library. Normally the technology library is known by the transistor size (eg. 65nm).

Usually, a circuit description is written in Hardware Description Language (HDL), such as Verilog. There are different topologies that can be applied for the same design. There are different approaches for designing a large system. One of the approaches is bottom up, i.e. start with the smallest module of the system and individually test them using test benches to make sure proper operation before integrating them into higher level modules. For example, in this part of the tutorial, we will build a 8-bit counter and simulate it. After proper operation is verified, we can create modules that will use this decoder, and so on.

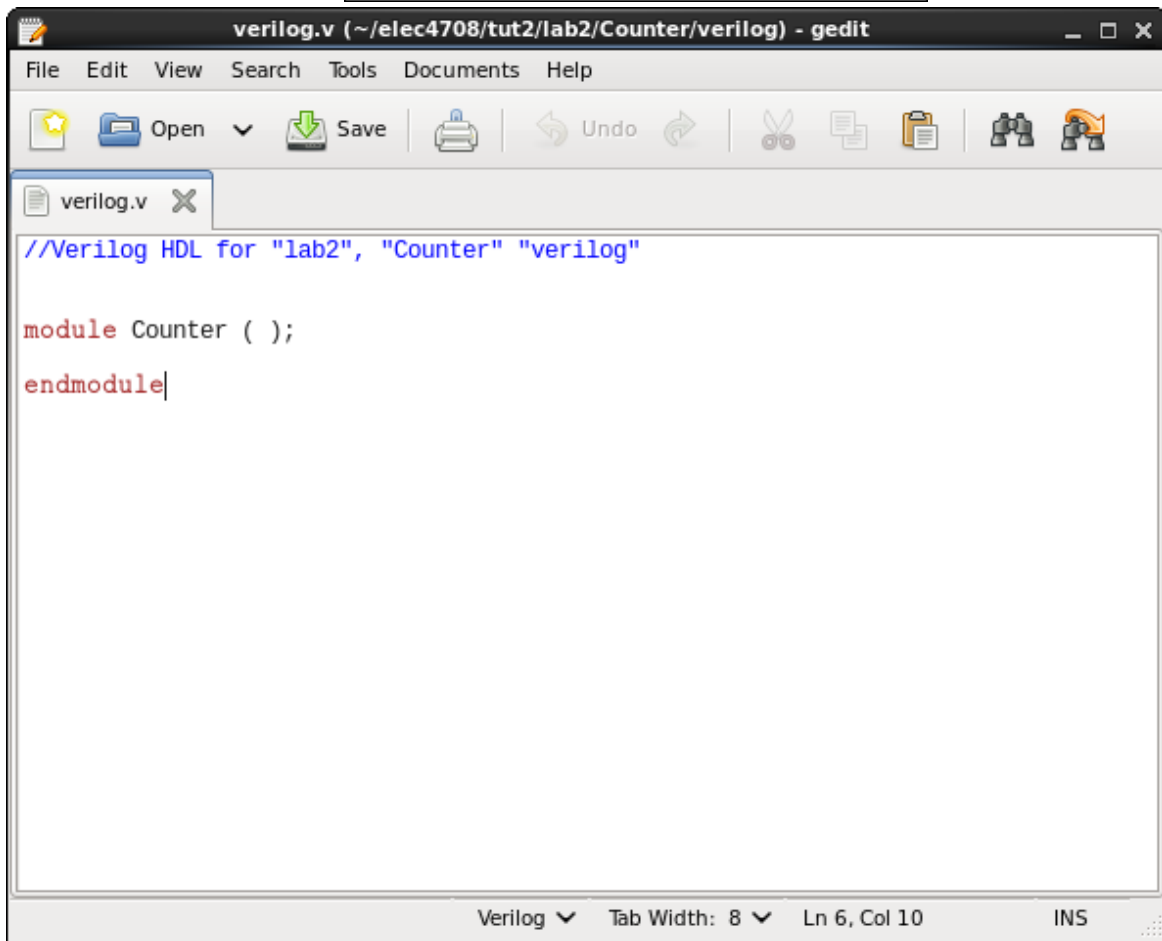
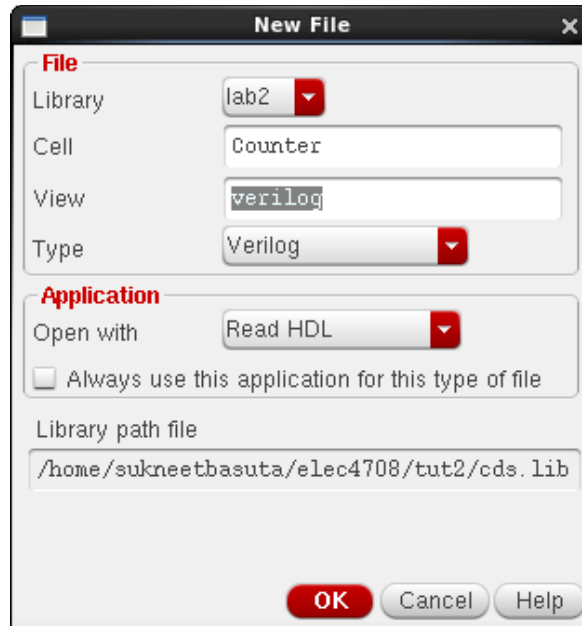
NOTE: The default editor on systems we are using is vim. If you are not comfortable with vi, or prefer something else, type export the environmental variable with you editor of choice before starting Cadence (i.e. "export EDITOR=nano"). If you don't know what vi is, use gedit. Before

1. Open a console and create a folder named "lab2" inside the elec4708 folder you created previously

```
> cd elec4708
> mkdir lab2
```
2. (Optional, but recommended) export your editor of choice (see above note)

```
> export EDITOR=gedit
```
3. Start Cadence and open the 65nm GP kit

```
> startCds
```
4. Open Library Manager and create a library named "lab2". (See tutorial for lab 1 part A for more details.)
5. Create a cell view named "Counter" inside the lab2 library. Select Application as "Read HDL", type as "Verilog", and view as "verilog". This will open a Verilog Editor as shown below. If you do not see it, double click on view->verilog (while lab2->Counter is highlighted.)



6. Type the following code for the 8-bit counter

If you forgot to change the editor or wish to use another one at this point, open the Verilog file in that application. The file should be located in `elec4708/lab2/lab2/Counter/verilog/verilog.v`

```

module Counter (up_down, enable, Clk, reset, out);
input up_down, enable, Clk, reset;

output [7:0] out;
reg [7:0] out;

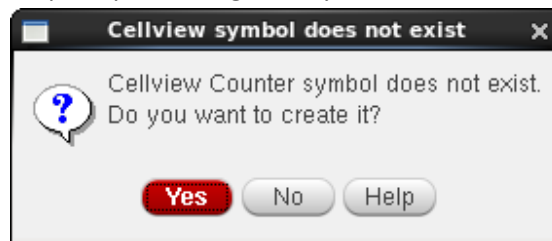
always @(posedge Clk or posedge reset)
    if (reset) begin
        out <= 8'b0;
    end else if (enable) begin
        if (up_down) begin
            out <= out + 1;
        end else begin
            out <= out - 1;
        end
    end
end

endmodule

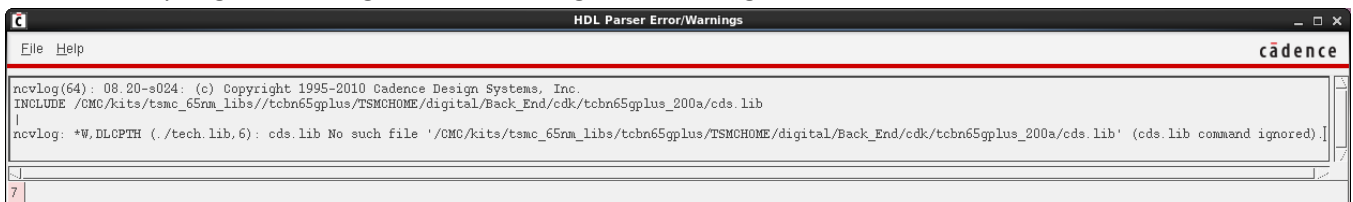
```

If you have trouble understanding the code, consult a verilog book or the guides linked to on the course website.

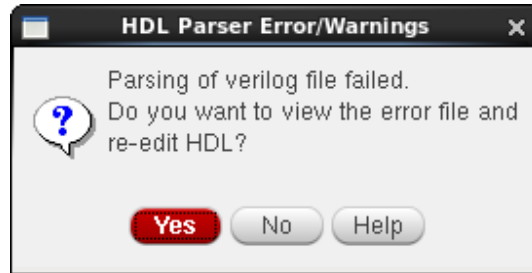
- When you quit the editor, a prompt will be given if you want to create a symbol. Select yes.



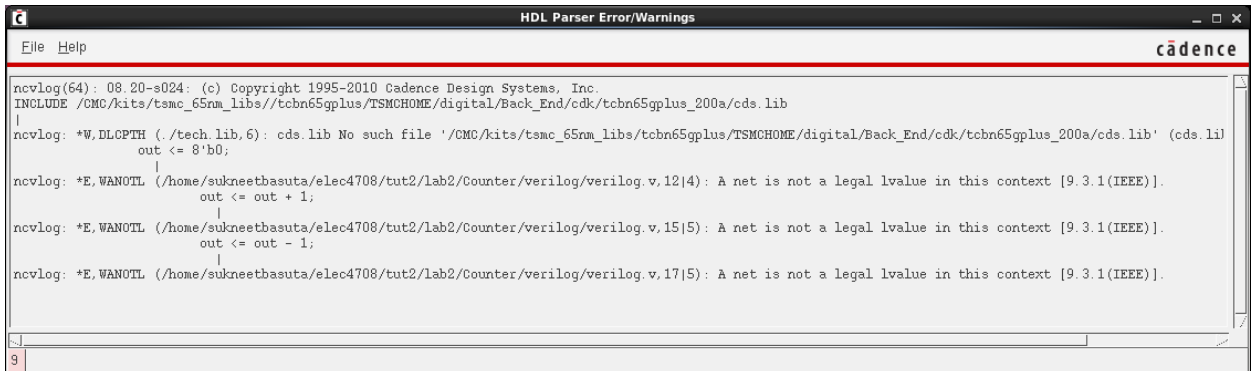
If you get a warning like the following, it is safe to ignore.



If the Verilog code has an error, you will receive a popup like the following.

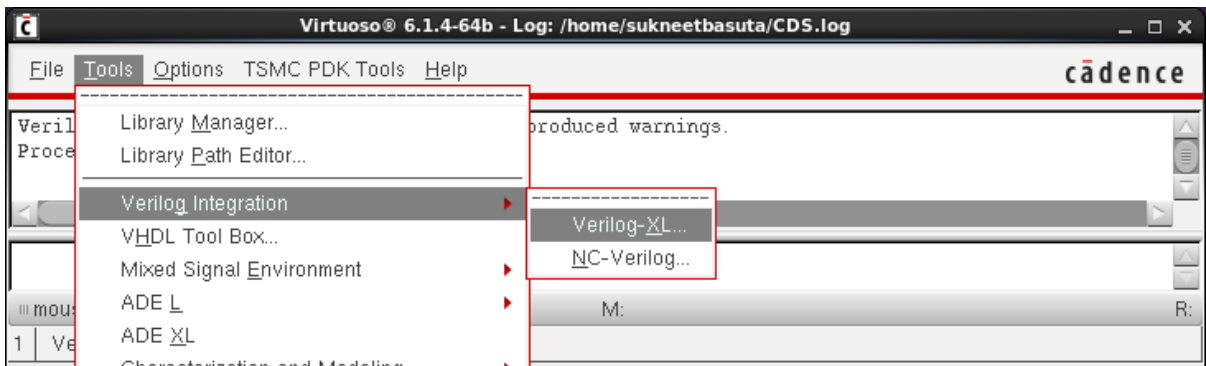


Hit yes, and a list of the errors will appear in a window like below.

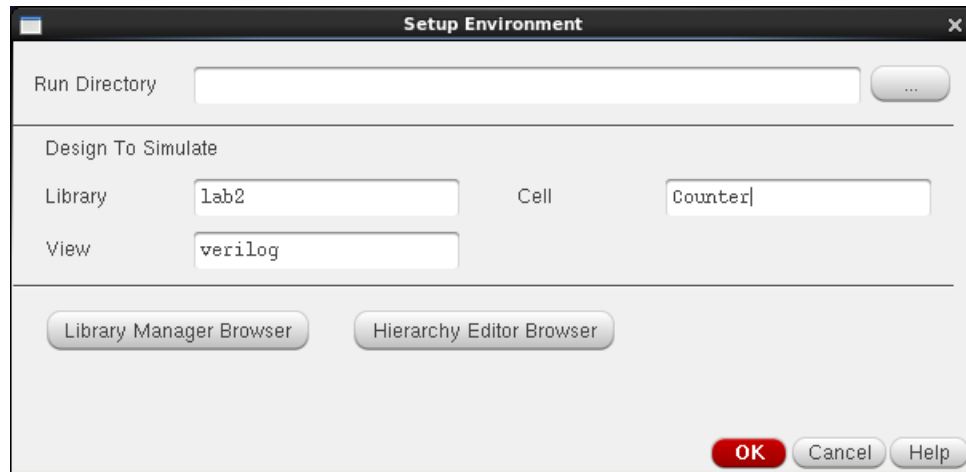


If you used an external editor, you might need to create a symbol if your simulation does not work. To create the symbol, follow the steps:

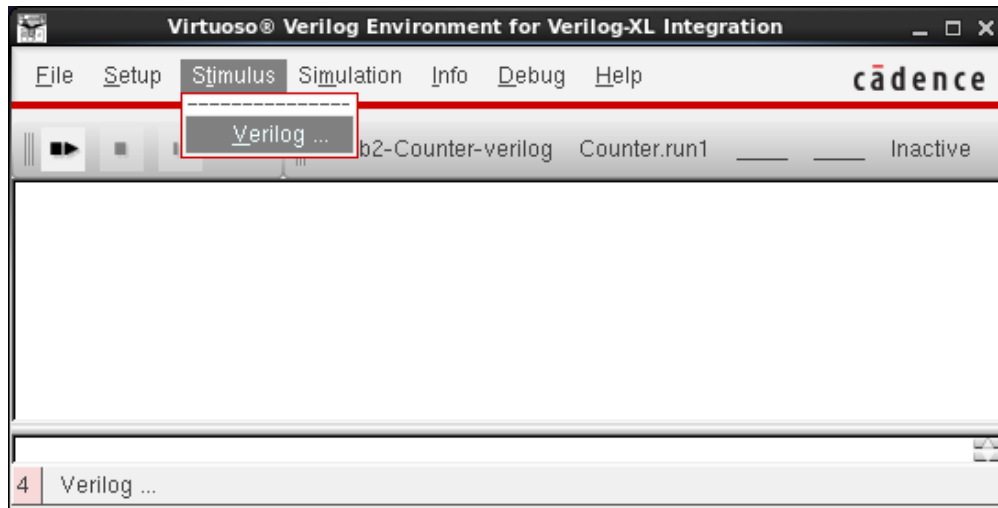
- a. Open Library Manager and select lab2->Counter->verilog. Double click on verilog to open the file. Press Enter in the editor window and you should be able to see your code.
 - b. Go down to a blank line using down arrow key and press “i” to start edit mode. Then press Enter to put a blank line. Press Esc to come out of edit mode. Type “:wq” to save and quit vi editor.
 - c. A prompt will be given if you want to generate a symbol. Choose YES.
8. Click **Tools->Verilog Integration->Verilog-XL** in Virtuoso console window



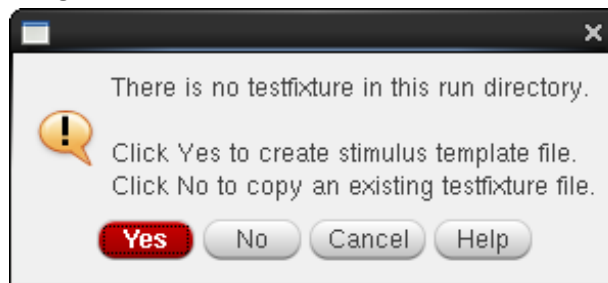
9. In the setup environment window, click “**Library Manager Browser**” and select your verilog code (Lab2->Counter->Verilog). Click close in the browser. Your “**Setup Environment**” window should look like this (it is okay to leave the run directory empty). Press OK.



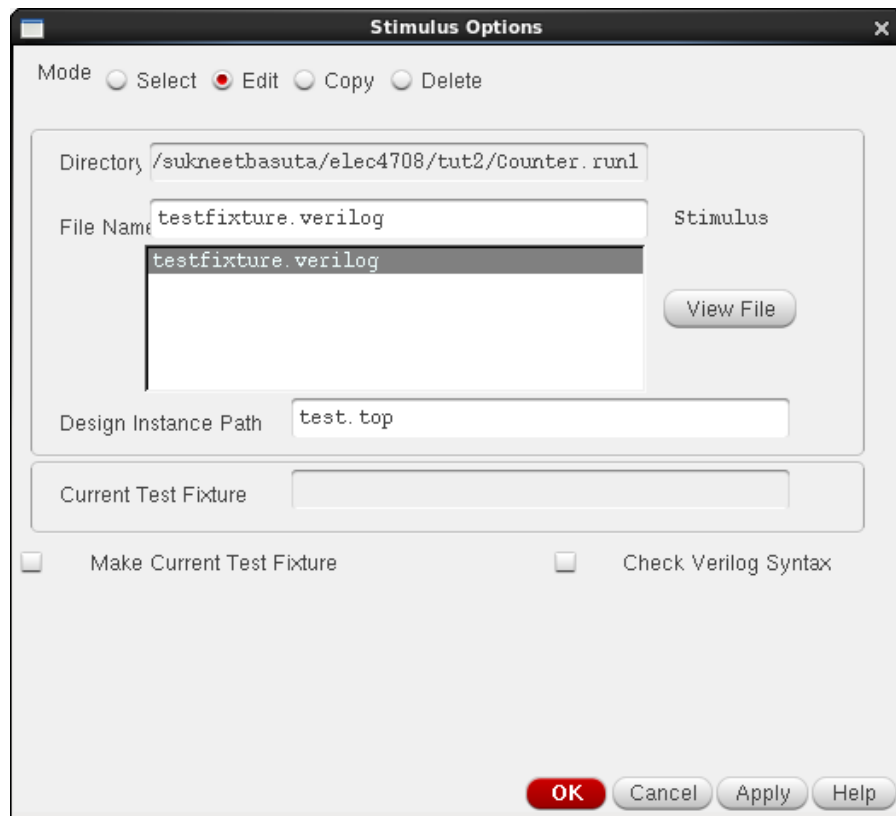
10. Virtuoso Verilog Environment for Verilog-XL Integration should open. Investigate different menu options and help files. When you are done, click on **Stimulus->verilog**.



11. A window will popup asking to create a new stimulus file. Hit **Yes**.



12. The Stimulus Option window will popup. Change the Mode to Edit. Select the stimulus file name in the list and hit OK.



13. The editor will popup with the stimulus file. Enter the following into the file. Make a backup copy of this testfixture in case the testfixture file is modified by the tool. If you wish to edit the file with another editor, the file is located in elec4708/lab2/Counter.run1/testfixture.verilog

```

always begin
    #5 Clk = ~Clk;
end

initial
begin
    Clk = 1'b0; enable = 1'b0; reset = 1'b1; up_down = 1'b0;
    #10 enable = 1'b1; reset = 1'b0; up_down = 1'b1;
    #10 up_down = 1'b1;
    #10 up_down = 1'b1;
    #10 up_down = 1'b1;
    #10 up_down = 1'b0;
    #10 up_down = 1'b0;
    #10 up_down = 1'b1;
    #10 up_down = 1'b1;
    #10 up_down = 1'b0;
    #10 $finish;
end

```

14. Click on **Start Interactive** button. The code should initialize. Then click **Continue** button to run the simulation. Once it's done, click on **View Waveforms** button.



Start Interactive

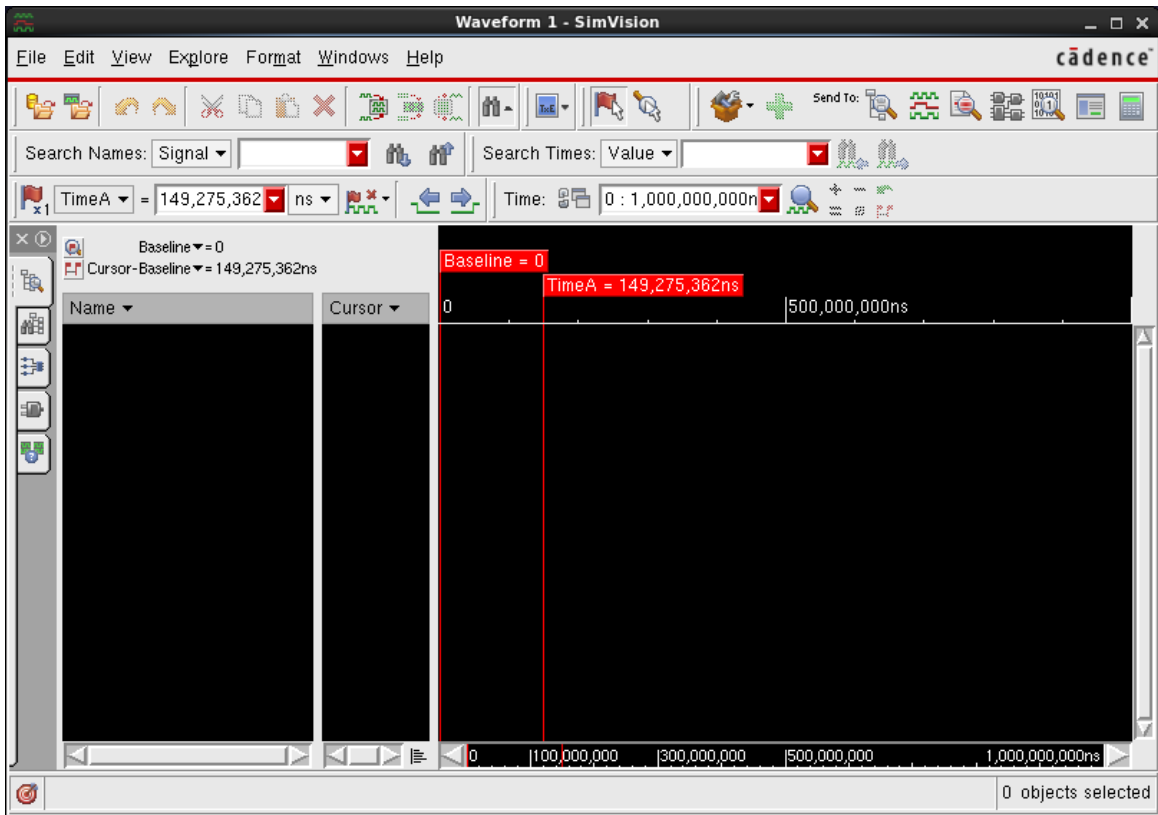


Continue




View Waveform

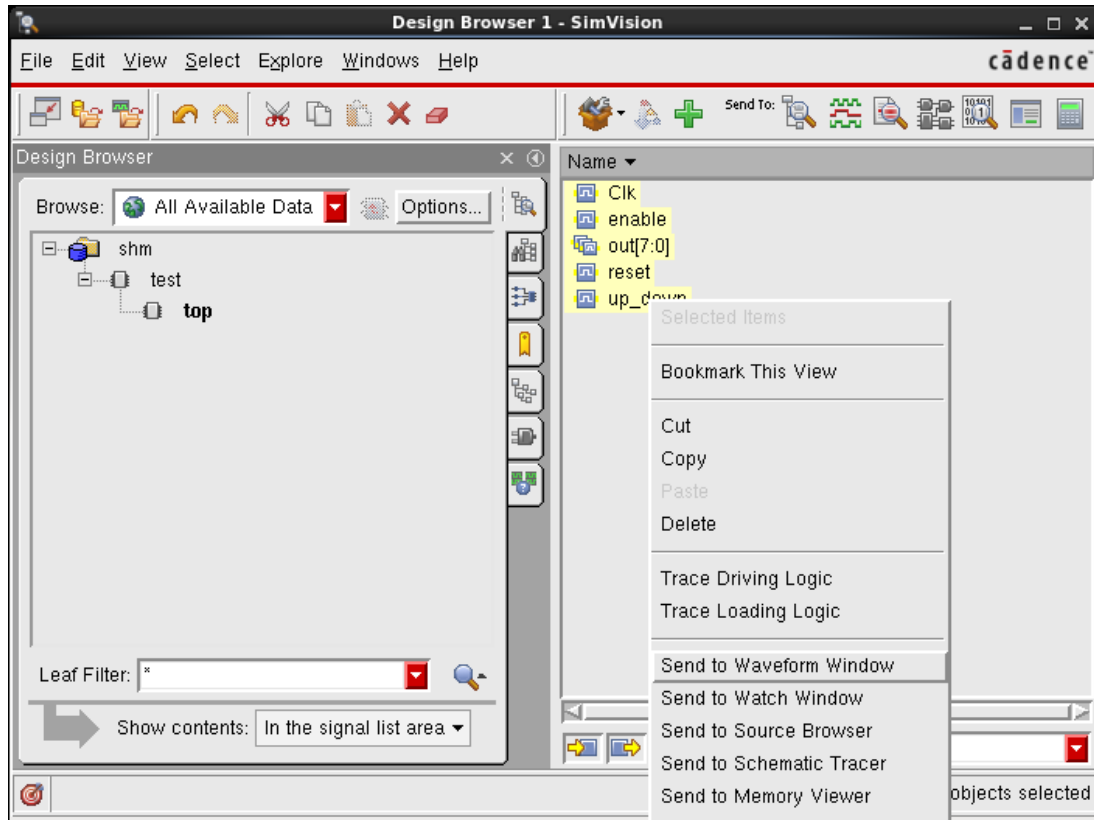
15. The waveform viewer window should appear. Investigate different menu options and panels of this window.



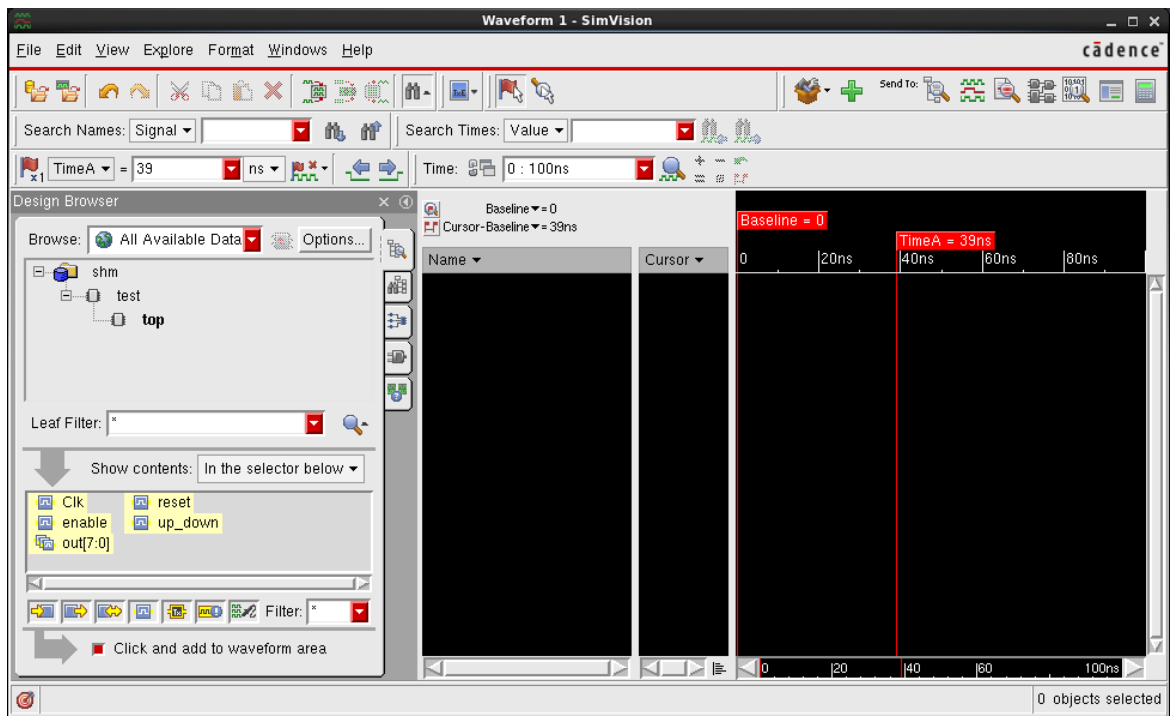
16. Click on **Windows-> New -> Design Browser**. The design browser window will appear. Again investigate menu options and buttons to familiarize you. Click on Shm->test->top on the left pane, then select all the signals (by dragging and selecting, or holding the ctrl or shift key). Then right click, select Send to Waveform Window (you have to hold and move mouse).



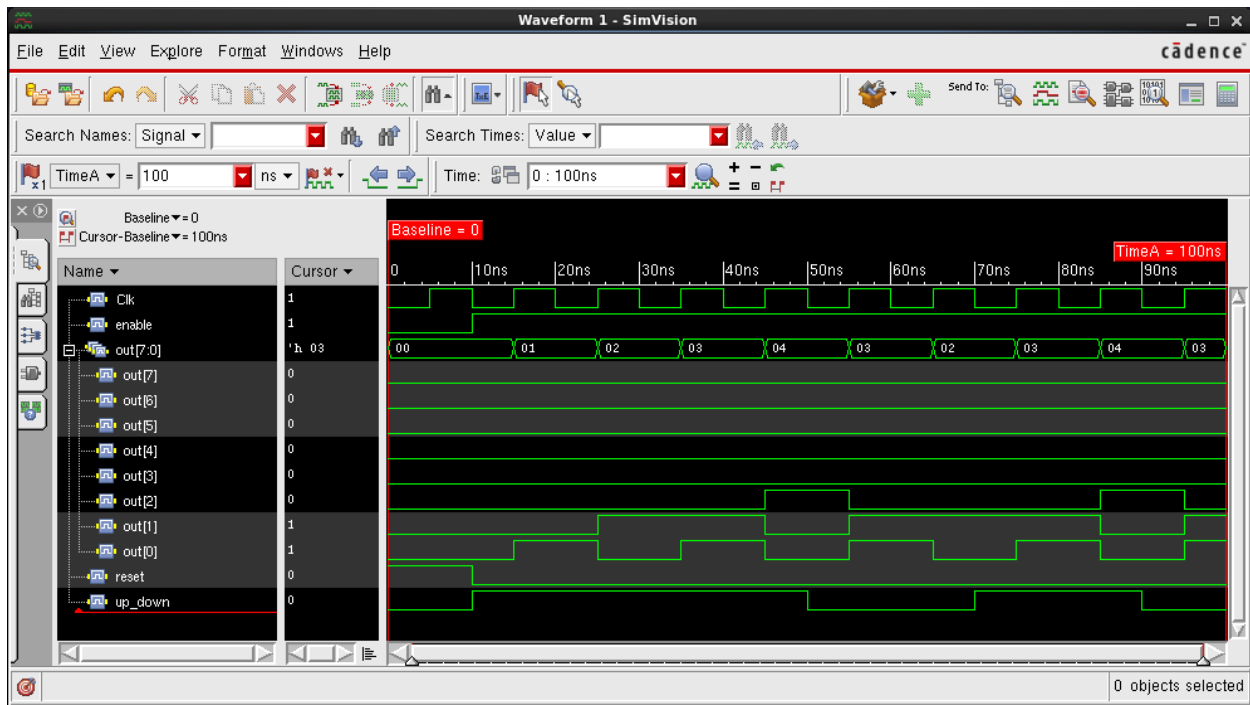
Alternatively select the Design Browser tab on the left , and left-click the signals you want to plot.



Or



- The traces should be seen in Waveform window. Double click on sel[2:0] and res[7:0] to expand them. Go through the waveform in different time slots and verify functionality of the decoder. You might want to “Zoom to X” to see whole waveform combinations.



When you are done, follow a similar procedure to write and simulate verilog code for the Shift-In/Out and 2's Complement to verify functionality of each module in the multiplier. If you do not follow this approach and your whole program does not work, it will be extremely difficult, if not impossible, to find out any logical mistake you make in your module level. In next part of the tutorial we will do a sample synthesis.