EECS 112L ORG DIGITL COMP LAB

Mentor Graphics QuestaSim Tutorial

Prepared by: Aniketh Esamudra Prakash

1. Follow the MobaXterm Tutorial (SSH_client.pdf) for connecting to the server and logging into your account.
2. Use Notepad++ to edit the files(.vhd or .sv format). Download Notepad++ for free from the below mentioned website

https://notepad-plus-plus.org/download/v6.8.3.html

Simulations in Mentor Graphics QuestaSim

In this tutorial, you will learn to compile, optimize and simulate the design. The design is written using the VHDL code. The testbench is in System Verilog. There are steps to simulate the VHDL design using System Verilog testbench.

How to Simulate the VHDL design using the System Verilog Testbench

The steps are explained by taking the example of the design file and testbench. alu_32.vhd is considered as an example for VHDL design. alu_32_tb.sv is considered as an example for System Verilog testbench. The same files are used to explain the steps.

A. Design the VHDL code for the 32 bit ALU

We are considering the VHDL alu_32.vhd file as an example. We could also do different design and play with the language.

VHDL Code for ALU Design

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ALU_32 is
  port
  (
    A, B     : in  std_logic_vector(31 downto 0);
    opcode   : in  std_logic_vector(3 downto 0);
    overflow : out std_logic;
    c_out    : out std_logic;
    equal    : out std_logic;
```

```
        output1  : out std_logic_vector(31 downto 0)
      );
    end entity ALU_32;

    architecture Behavioral of ALU_32 is
      signal temp: std_logic_vector(32 downto 0);
    begin
      process(A, B, opcode, temp) is
      begin
        overflow <= '0';
    case opcode is

         when "0000" =>  -- No operation
                 output1    <= (others => '0');
                 c_out     <= '0';
                 overflow   <= '0';
                 equal     <= '0';

         when "0001" => -- ADD
                 Temp     <= std_logic_vector((unsigned("0" & A) + unsigned(B)));
                 output1  <= temp(31 downto 0);
                 c_out    <= temp(32);
                 overflow <= temp(32);
                 equal <= '0';

         when "0010" => -- SUB
                 if (A >= B) then
                         output1 <= std_logic_vector(unsigned(A) - unsigned(B));
                         overflow   <= '0';
                         equal <= '0';
                 else
                         output1 <= std_logic_vector(unsigned(B) - unsigned(A));
                         overflow   <= '1';
                         equal <= '0';
                 end if;

        when "0011" => -- COMP
                if (A = B) then
                         equal <= '1';
                         c_out     <= '0';
                         overflow   <= '0';
                else
```

```
                        equal <= '0';
                        c_out     <= '0';
                        overflow   <= '0';
                end if;

        when "0101" => -- AND
                output1 <= A and B;
                equal <= '0';
                c_out     <= '0';
                overflow   <= '0';

        when "0110" => -- OR
                output1 <= A or B;
                equal <= '0';
                c_out     <= '0';
                overflow   <= '0';

        when "0111" => -- NOT
                output1 <= not A;
                equal <= '0';
                c_out     <= '0';
                overflow   <= '0';

        when "1000" => -- XOR
                output1 <= A xor B;
                equal <= '0';
                c_out     <= '0';
                overflow   <= '0';

        when "1001" => -- SLL
                output1 <= std_logic_vector(unsigned(A) sll
        to_integer(unsigned(B)));
                equal <= '0';

        when "1011" => -- MOV
                output1 <= A;
                equal <= '0';
        when others =>
                output1    <= (others => '0');
                c_out     <= '0';
                overflow   <= '0';
                equal     <= '0';
```

```
    end case;
  end process;

end architecture Behavioral;
```

a. Open the notepad++ tool.
b. Design using VHDL code.
c. Save the file using .vhd format. (Example: alu_32.vhd)
d. Upload the file to your account in the server using MobaXterm

B. Design the Testbench using System Verilog Language

We will design the Testbench using the System Verilog Language. We are considering the alu_32_tb.sv as an example.

System Verilog Testbench for testing alu_32_tb.sv

```
module alu_32_tb;
  logic[31:0] A;
  logic[31:0]  B;
  logic[3:0] opcode;
  wire overflow;
  wire c_out;
  wire equal;
  wire[31:0] output1;

alu_32 L1(
        .A(A)
       ,.B(B)
       ,.opcode(opcode)
       ,.overflow(overflow)
       ,.c_out(c_out)
       ,.equal(equal)
       ,.output1(output1)
       );

initial begin
/*arithmetic operation*/
        A = 32'b01010101010101010101010101010101;
         B = 32'b10101010101010101010101010101010;
                     opcode = 4'b0000;
                   #100;
                   opcode = 4'b0001;/*-- ADD*/
                   #100;
                   opcode = 4'b0010;/*--SUB*/
                   #100;
                   opcode = 4'b0011;/*--COMP*/
                   #100;
                   opcode = 4'b0101;/*--AND*/
                   #100;
                   opcode = 4'b0110; /*--OR*/
               #100;
                   opcode = 4'b0111;/*--NOT*/
                  #100;
                   opcode = 4'b1000;/*--XOR*/
                   #100;
                   opcode = 4'b1001;/*--SLL*/
                   #100;
                   opcode = 4'b1011; /*--MOV*/
                   #100;

     /*round 2*/
       A = 32'b10101010101010101010101010101010;
       B = 32'b01010101010101010101010101010101;
                  opcode = 4'b0000;
                  #100;
                   opcode = 4'b0001;/*-- ADD*/
                   #100;
                   opcode = 4'b0010;/*--SUB*/
                   #100;
                   opcode = 4'b0011;/*--COMP*/
                   #100;
                   opcode = 4'b0101;/*--AND*/
                   #100;
                   opcode = 4'b0110; /*--OR*/
                 #100;
                   opcode = 4'b0111;/*--NOT*/
                 #100;
                   opcode = 4'b1000;/*--XOR*/
                   #100;
                   opcode = 4'b1001;/*--SLL*/
                   #100;
```

```
                    opcode = 4'b1011; /*--MOV*/
                    #100;

        /*round 3*/
            A = 32'b00101010101110111010101011111110;
           B = 32'b00101010101110111010101011111110;
                   opcode = 4'b0000;
                   #100;
                    opcode = 4'b0001;       /*-- ADD*/
                    #100;
                    opcode = 4'b0010;       /*--SUB*/
                    #100;
                    opcode = 4'b0011;       /*--COMP*/
                    #100;
                    opcode = 4'b0101;       /*--AND*/
                  #100;
                    opcode = 4'b0110; /*--OR*/
                   #100;
                    opcode = 4'b0111;/*--NOT*/
                   #100;
                    opcode = 4'b1000;/*--XOR*/
                    #100;
                    opcode = 4'b1001;/*--SLL*/
                    #100;
                    opcode = 4'b1011; /*--MOV*/
                    #100;

        /*round 4*/ /* Test SLL */
              A = 32'b00101010101110111010101011111110;
              B = 32'b00000000000000000000000000000011;
                   opcode = 4'b1001;
                    #100;

        end
    endmodule
```

a. Now, design the testbench using the system verilog code using notepad++.
b. Save the file using .sv format. (Example: alu_32_tb.sv)
c. Upload the file to your account in the server using MobaXterm

C. <u>Copy the below mentioned files to your account in the server</u>
   Make few modifications to these files before uploading them to your online account in the server.

The modifications are mentioned below

- In ***rtl.cfg*** file, modify ***dut.v*** to ***alu_32.vhd***
- In ***tb.cfg*** file, modify ***tb_top.sv*** to ***alu_32_tb.sv***
- In ***sim.do*** file, modify the command ***log -r tb_top.**** to ***log -r alu_32_tb.****

After making the above mentioned modifications in the corresponding files, upload all the below mentioned files to the same location in the server where the design and testbench were uploaded before.

1. **pre_compile.csh**
2. **setup.csh**
3. **rtl.cfg**
4. **tb.cfg**
5. **sim.do**

D. Go to the folder location in your online account in the server, where the files are uploaded. Use the following Linux commands to do the compilation, optimization, simulation and to view the waveform.
   Note: All these Linux commands should be executed in the folder location where these files are uploaded in the server.

1. ***source setup.csh***
   This source command will load the file into the command prompt. It reads and executes the commands from the file ***setup.csh***

2. ***source pre_compile.csh***
   This source command will load the file into the command prompt. It reads and executes the commands from the file ***pre_compile.csh***

3. **Compile the VHDL design file: alu_32.vhd**
   Run ***vcom -64 -f rtl.cfg*** on the command line.
   Here,

✓ ***vcom*** is a command which compiles the VHDL source code ***alu_32.vhd*** into a work library by default
✓ ***-64*** represents that vcom uses 64-bit executable
✓ ***-f*** specifies the argument file with command lines arguments, which allows to use the complex arguments to be used once again without retyping.
✓ ***rtl.cfg*** is the file which has the name of the VHDL file: ***alu_32.vhd***

4. **Compile the System Verilog file: alu_32_tb.sv**
   Run ***vlog -64 -sv -f tb.cfg*** on the command line.
   Here,

✓ ***vlog*** is a command which compiles the Verilog source code/System Verilog Extensions (Here it is ***alu_32_tb.sv***)
✓ ***-64*** represents that vlog uses 64-bit executable
✓ ***-f*** specifies the argument file ***tb.cfg*** , which allows to use the complex arguments to be used once again without retyping.
✓ ***-sv*** is used to enable the System Verilog features and keywords
✓ ***tb.cfg*** is the file which has the name of the System Verilog Testbench: ***alu_32_tb.sv***

5. **Optimize the VHDL design: alu_32.vhd**
   Run ***vopt -64 tb_top +acc=mpr -o tb_top_opt*** on the command line
   Example: ***vopt -64 alu_32_tb +acc=mpr -o alu_32_tb_opt***
   in this example
   Here,

✓ ***vopt*** is used to do the global optimization on the design after the compilation has been done using ***vcom*** or ***vlog***
✓ ***-64*** represents that vopt uses 64-bit executable
✓ ***+acc*** provides visibility into the design for debugging purposes
✓ ***-o*** allows us to designate the name of the optimized design file ***alu_32_tb_opt***

6. **Simulate the VHDL Design: alu_32.vhd using System Verilog Testbench: alu_32_tb.sv**
   Run ***vsim -64 -l simulation.log -do sim.do -c tb_top_opt*** on the command line.
   Example: ***vsim -64 -l simulation.log -do sim.do -c alu_32_tb_opt*** in our alu_32 example.
   Here,

✓ ***vsim*** command is used to simulate the VHDL design or an entity/architecture pair or a verilog module/system verilog extension or an optimized design.
✓ ***-64*** represents that vsim uses 64-bit executable
✓ ***-l simulation.log*** overrides the transcript file with the *modelsim.ini* variables.
✓ ***-do sim.do*** tells the vsim to use the commands specified in the do file.
✓ ***-c*** specifies that the simulator will run in command-line mode

7. View the Waveform

- ✓ Run *vsim -view waveform.wlf*

  Here,
- ✓ *vsim* is also used to view the results of the previous simulation run when *-view* switch is invoked
- ✓ *waveform.wlf* is the simulation file. It is opened to view the waveform.

After running the command, a window will pop up. That window is shown below in the Figure 1

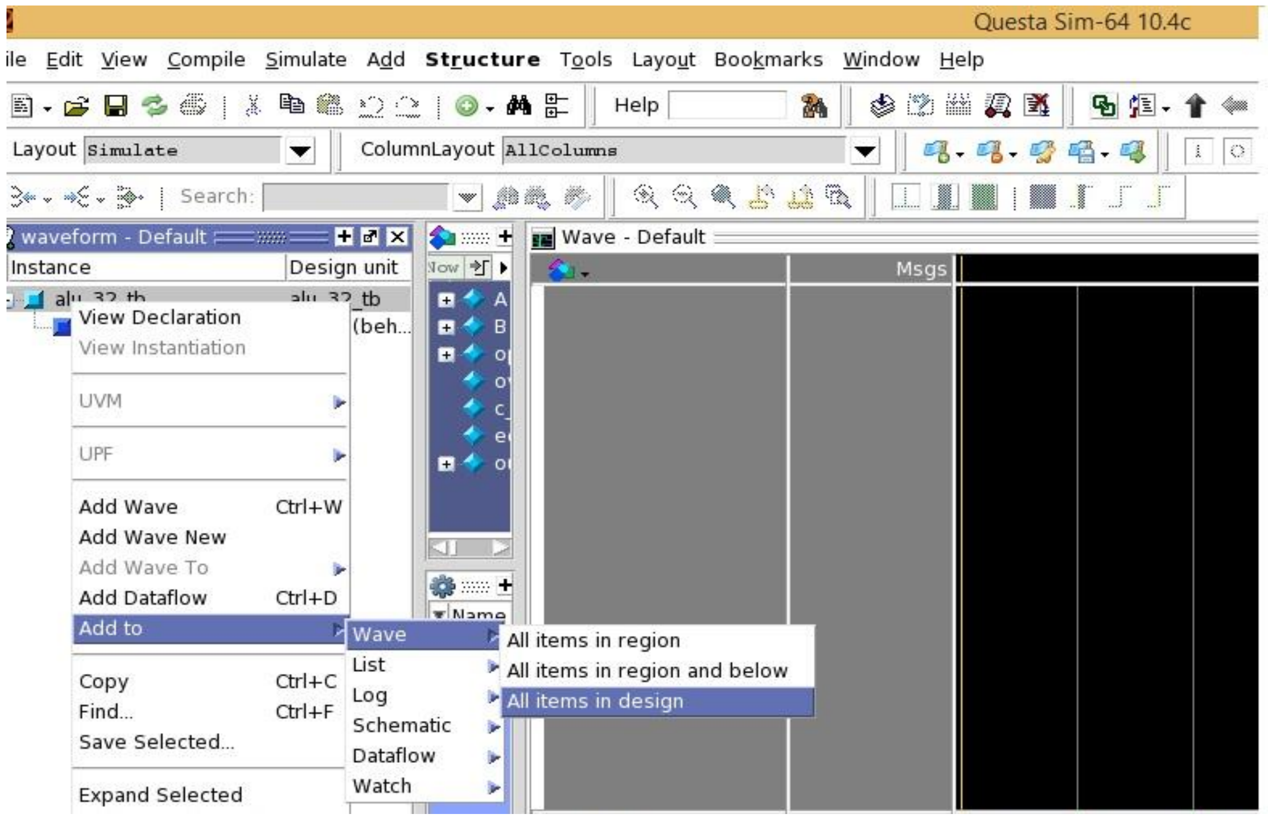
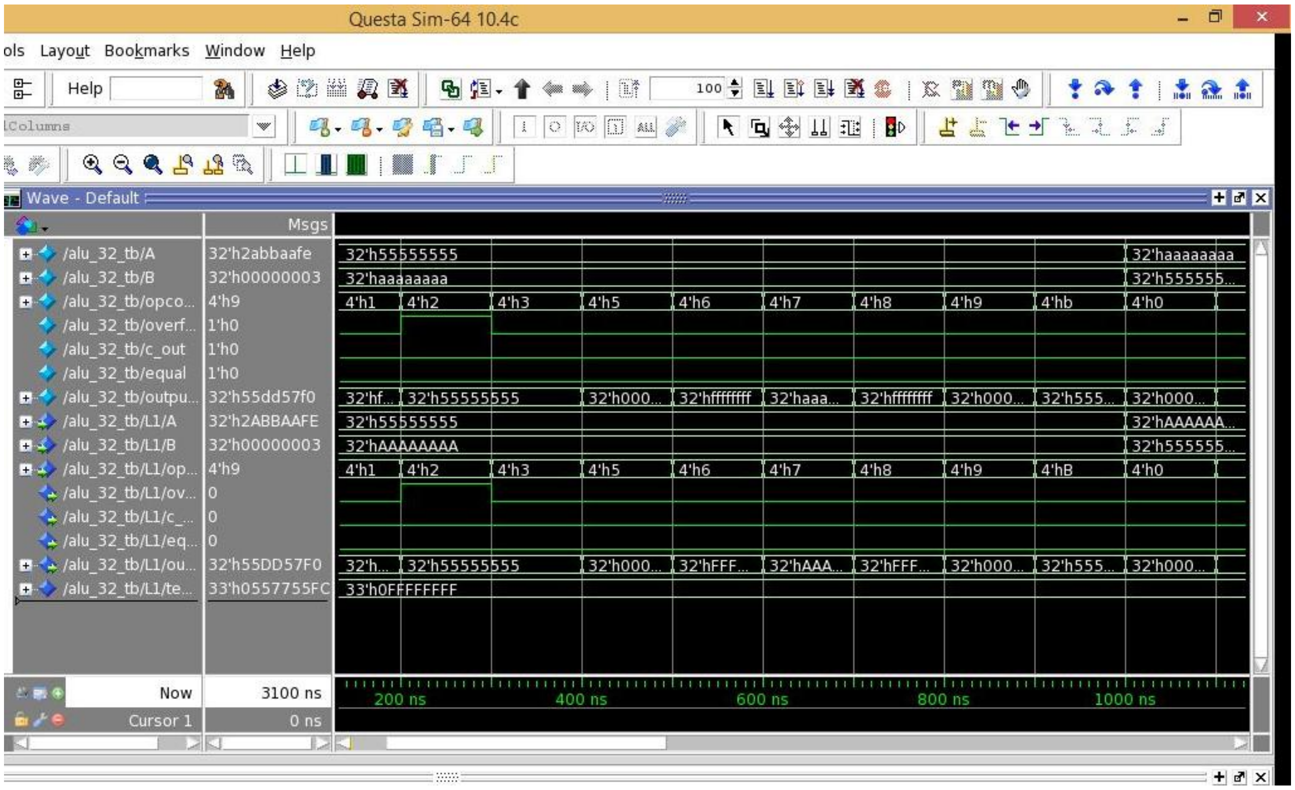


Figure 1. Step to add waveforms in the simulation window

- ✓ Use the mouse and do the right click on the **alu_32_tb** in the waveform-default tab.
- ✓ Move the mouse pointer on the **Add to** and do the left click. Then move the mouse pointer on **Wave** and on **All Items in Design** and select **All Items in Design.**

- ✓ Simulated waveforms should appear on the right side of the screen. It is shown in Figure 2 below.



```
/alu_32_tb/A          32'h2abbaafe
/alu_32_tb/B          32'h00000003
/alu_32_tb/opco...    4'h9
/alu_32_tb/overf...   1'h0
/alu_32_tb/c_out      1'h0
/alu_32_tb/equal      1'h0
/alu_32_tb/outpu...   32'h55dd57f0
/alu_32_tb/L1/A       32'h2ABBAAFE
/alu_32_tb/L1/B       32'h00000003
/alu_32_tb/L1/op...   4'h9
/alu_32_tb/L1/ov...   0
/alu_32_tb/L1/c_...   0
/alu_32_tb/L1/eq...   0
/alu_32_tb/L1/ou...   32'h55DD57F0
/alu_32_tb/L1/te...   33'h0557755FC
Now                   3100 ns
Cursor 1              0 ns
```

```
A:       32'h55555555 | 32'haaaaaaaa
B:       32'haaaaaaaa | 32'h555555...
opcode:  4'h1 | 4'h2 | 4'h3 | 4'h5 | 4'h6 | 4'h7 | 4'h8 | 4'h9 | 4'hb | 4'h0
output:  32'hf... | 32'h55555555 | 32'h000... | 32'hffffffff | 32'haaa... | 32'hffffffff | 32'h000... | 32'h555... | 32'h000...
L1/A:    32'h55555555 | 32'hAAAAAA...
L1/B:    32'hAAAAAAAA | 32'h555555...
L1/op:   4'h1 | 4'h2 | 4'h3 | 4'h5 | 4'h6 | 4'h7 | 4'h8 | 4'h9 | 4'hB | 4'h0
L1/ou:   32'h... | 32'h55555555 | 32'h000... | 32'hFFF... | 32'hAAA... | 32'hFFF... | 32'h000... | 32'h555... | 32'h000...
L1/te:   33'h0FFFFFFFF
```

Figure 2. Simulated Waveforms

- ✓ Check the waveforms based on the desired functionality. If the waveforms do not follow the desired functionality, then make the changes in the design or testbench. Do the compilation and further steps which follows.

If you get the desired results and functionality, then your design is good enough to be proud of your efforts!