CPE 300L
DIGITAL SYSTEM ARCHITECTURE AND DESIGN LABORATORY

LABORATORY 3
DESIGN AND TESTING OF COMBINATIONAL CIRCUITS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF NEVADA, LAS VEGAS

OBJECTIVE

Get familiar with Modelsim 10.1d software and review the design of combinational circuits.

INTRODUCTION

ModelSim eases the process of finding design defects with an intelligently engineered debug environment.

The ModelSim debug environment efficiently displays design data for analysis and debug of all languages. ModelSim allows many debug and analysis capabilities to be employed post-simulation on saved results, as well as during live simulation runs.

Features of ModelSim:

- Unified mixed language simulation engine for ease of use and performance
- Native support of Verilog, SystemVerilog for design, and VHDL, for effective verification of sophisticated design environments
- Fast time-to-debug, easy to use, multi-language debug environment
- Advanced code coverage and analysis tools for fast time to coverage closure
- Interactive and Post-Sim Debug available so same debug environment used for both
- Powerful Waveform Compare for easy analysis of differences and bugs
- Advanced code coverage and analysis tools for fast time to coverage closure
- Unified Coverage Database with complete interactive and HTML reporting and processing for understanding and debugging coverage throughout your project
- Assertions
- Gain insight and visibility into your design by letting assertions notify you of an error so it can be fixed
- ModelSim provides a powerful library of checkers (OVL) letting you debug with assertions right away, without writing your own
- Assertions can also serve as documentation for your design, as comments are embedded into the code as you go
**THEORY OF OPERATION**

1. **Megafunctions**
Altera megafunctions are complex or high-level building blocks that can be used together with gate and flipflop primitives in Quartus II design files. The parameterizable megafunctions and LPM functions provided by Altera are optimized for Altera device architectures. You must use megafunctions to access some Altera device-specific features, such as memory, DSP blocks, LVDS drivers, PLLs, and SERDES and DDIO circuitry.

The lpm_mult megafunction is one of the arithmetic megafunctions provided in the Quartus® II software MegaWizard® Plug-In Manager. The basic function of a multiplier is to multiply two input data values to produce a product as an output. Figure 1 shows a basic multiplier.

![Multiplier Diagram](image)

*Fig. 1. Multiplier Megafuction*

2. **Testbench**
Once a Verilog model for a system has been made, the next step is to test it. A model has to be tested and validated before it can be successfully used. A *testbench* is a piece of Verilog code that can provide input combinations to test a Verilog model for the system under test. It provides stimuli to the system or circuit under test. Testbenches are frequently used during simulation to provide sequences of inputs to the circuit or Verilog model. Below – an example for a 2-input AND gate.

Verilog code for AND gate:

```verilog
module and_gate (  
    input a,  
    input b,  
    output out  
);  
    assign out = a & b;  
endmodule
```
Having a code, the associated testbench can be created. Testbench 1 contains just the signal declarations:

Step 1: signal declarations

```vhdl
01 module and_tb;
02   reg a, b;
03   wire out;
04
05   and_gate U0 (  
06     .a   (a),
07     .b   (b),
08     .out (out)  
09  );
10 endmodule
```

Step 2: Adding the simulation data

```vhdl
01 module and_tb;
02   reg a, b;
03   wire out;
04
05   and_gate U0 (  
06     .a   (a),
07     .b   (b),
08     .out (out)  
09  );
10
11 initial begin
12   $dumpfile ("and_gate.vcd");
13   $dumpvars;
14   a = 0;
15   b = 0;
16   #10
17   a = 0;
18   b = 1;
19   #10
20   a = 1;
21   b = 0;
22   #10
23   a = 1;
24   b = 1;
25   #10
26   a = 0;
27   b = 0;
28   $finish;
29 end
30 endmodule
```
This form of a testbench can be executed. Testbench result:

![Waveform](image.png)

Fig. 2. Testbench result waveform

The output of the testbench can be also done in a form of value series. Consider the following code (lines 11-14 added, comparing to the previous code):

```verilog
module and_tb;
reg a, b;
wire out;
and_gate U0 (.a (a), .b (b), .out (out));
initial begin
$display("time\ta,\tb,\tout");
$monitor("%d,\ta,\tb,\tout", $time, a, b, out);
end
initial begin
$dumpfile ("and_gate.vcd");
$dumpvars;
a = 0;
b = 0;
#10
a = 0;
b = 1;
#10
a = 1;
b = 0;
#10
a = 1;
b = 1;
#10
a = 0;
b = 0;
#10
a = 0;
b = 1;
#10
a = 1;
b = 1;
#10
a = 0;
b = 0;
$finish;
end
endmodule
```
This code will yield both `and_gate.vcd` file (used for waveforms) and the series of simulation values:

```
Compiler version J-2014.12-SP1-1; Runtime version J-2014.12-SP1-1;
time, a, b, out
0, 0, 0, 0
10, 0, 1, 0
20, 1, 0, 0
30, 1, 1, 1
40, 0, 0, 0
$finish called from file "testbench.sv", line 33.
$finish at simulation time 50
```

**LAB DELIVERIES**

**PRELAB**

1. **4-to-1 MUX in Verilog**
   Write your own 4-to-1 MUX in Verilog and simulate in Quartus. Include the simulation results in your prelab.

2. **Decoder in Verilog**
   Write the decoder in Verilog using the truth table below. Simulate in Quartus. Include the simulation results in your prelab.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>y₁</th>
<th>y₂</th>
<th>y₃</th>
<th>y₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**LAB EXPERIMENTS**

1. **Complete the Altera Modelsim tutorial:**
   [https://faculty.unlv.edu/eelabs/docs/labs/cpe300L/files/01/lab_01_modelsim_tutorial.pdf](https://faculty.unlv.edu/eelabs/docs/labs/cpe300L/files/01/lab_01_modelsim_tutorial.pdf)

   1. **Equality Comparator**
      This comparator compares two 4-bit numbers for equality.
      - Design the comparator and implement in Verilog.
      - Write the testbench.
      - Simulate in Modelsim and obtain waveforms.
      - Upload to DE2 and test the design onboard.
2. Magnitude Comparator
This comparator compares two 1-bit numbers for magnitude. Three outputs: a>b, a=b, a<b.
- Design the comparator and implement in Verilog.
- Write the testbench.
- Simulate in Modelsim and obtain waveforms.
- Upload to DE2 and test the design onboard.

3. Ripple Adder
Design the 4-bit ripple adder.
- Design the adder and implement in Verilog.
- Write the testbench.
- Simulate in Modelsim and obtain waveforms.
- Upload to DE2 and test the design onboard.

4. Parametrized Megafuntions
Go through the Megafuntions LPM_MULT tutorial and complete the example.
https://faculty.unlv.edu/eelabs/docs/labs/cpe300L/files/03/lab_03_lpm_mult.pdf
- Obtain simulation waveforms through Modelsim

POSTLAB REPORT:

Include the following elements in your postlab report:
1. Theory of operation
   a. Describe what the megafuntion is
   b. List the applications of megafuntions
2. Your prelab report – include as a section 2 of the postlab
3. Results of the experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Delivery</th>
</tr>
</thead>
</table>
| 1          | a. Verilog code (paste in the report)  
|            | b. Testbench code (paste in the report)  
|            | c. Waveforms from ModelSim  
|            | d. ModelSim Project (zip file or link to Google Drive)  |
| 2          | a. Verilog code (paste in the report)  
|            | b. Testbench code (paste in the report)  
|            | c. Waveforms from ModelSim  
|            | d. ModelSim Project (zip file or link to Google Drive)  |
| 3          | a. Verilog code (paste in the report)  
|            | b. Testbench code (paste in the report)  
|            | c. Waveforms from ModelSim  
|            | d. ModelSim Project (zip file or link to Google Drive)  |
| 4          | a. Waveforms for ModelSim  
|            | b. ModelSim Project (zip file or link to Google Drive)  |

4. Answer the questions:
   a. Why ModelSim is better than Quartus internal simulator? (elaborate)
b. What are the other simulators for digital logic? List 3.

5. Conclusions
   a. Write down your conclusions, things learned, problems encountered during the lab and how they were solved, etc.