

CPSC 321:501-503 - Computer Architecture
Texas A&M University
Department of Computer Science

Fall 2006

Lab 5 - Introduction to Verilog (100 pts)
Complete by yourself

Release date: 16 October 2006
Due date: One week after the lab

Objective

In this lab, you will be introduced to Verilog. You will work on structural designs and specifications.

A simple example

Implementing a simple arbitrary Boolean function

Using primitive Verilog gates, implement the Boolean function: $ab + a'$. Name this module `first_module`. Write a test bench - `test_first` - to examine the outputs for varying input combinations. Compile the code with VCS and run the executable.

Here is what your Verilog code would look like:

```
module first_module(out, a, b);
    input    a, b;
    output   out;
    wire     a1, a2;

    not      n1(a1, a);
    and      and1(a2, a, b);
    or       or1(out, a1, a2);
endmodule

module test_first();
    reg      a, b;
    wire     out;

    first_module fm(out, a, b);

    initial begin
        $monitor ($time, "\ta=%b\tb=%b\tout=%b", a, b, out);
        a = 0; b = 0;
    end
endmodule
```

```

        #1 a = 0; b = 1;
        #1 a = 1; b = 0;
        #1 a = 1; b = 1;
    end

endmodule

```

Enter this code into your text editor (vi or emacs). Compile this code with VCS (Look at the VCS help document for further instructions).

Assignment

1. [10 pts] Half adder

A half-adder is a combinational circuit that adds two 1-bit inputs **a** and **b**, and generates 1-bit sum **s** and carry out **c**. The truth-table for a half-adder is shown below:

<i>a</i>	<i>b</i>	<i>s</i>	<i>c</i>
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Structurally define a half-adder. Prepare a test bench to test the half-adder and evaluate with VCS. Test this module completely, since you will be using it as a building block for the full-adder.

2. [10 pts] Full adder

Structurally define a 1-bit full adder using two half adders (from above). The truth table for the full-adder is shown below:

<i>a</i>	<i>b</i>	<i>carryIn</i>	<i>sum</i>	<i>carryOut</i>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Prepare a test bench to test the full-adder and evaluate with VCS. Test this module completely, since you will be using it as a building block for your multi-bit adders.

3. [25 pts] Multi-bit full adder

Your designs so far have been single bit with 1-bit ports, wires and registers. To model multi-bit objects, you may specify the size when declaring the object. For example, a 4-bit wide wire can be declared as:

```
wire [3:0] a;
```

The same may be done for registers and ports. You can also access the individual bit lines or a sub-group of the bit-lines.

Develop a 2-bit and 4-bit adder using multi-bit declarations and the 1-bit full-adder you developed earlier. Prepare a test bench and evaluate with VCS.

4. [25 pts] Decoders and multiplexers

a. Design and structurally define in Verilog a *5x32 decoder*. Prepare the test bench and evaluate the design with VCS.

b. Design and structurally define in Verilog a *32x1 multiplexer*. Prepare the test bench and evaluate the design with VCS.

5. [30 pts] 32-bit Ripple carry adder/subtractor

Design and structurally define a 32-bit adder/subtractor. You may use 1-bit or 4-bit adders to build your design. Prepare the test bench and evaluate the design with VCS.

References

1. [Verilog introduction](#)
 2. [Synopsys VCS usage instructions](#)
 3. [A note on the always construct](#)
-