Lab 6 (All Sections) Prelab: Introduction to Verilog

Name:

Sign the following statement:

On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work

1 Objective

The main objective of this lab is to give hands on experience with Verilog HDL and Synopsys VCS. For this lab you are expected to know some basic Verilog programming and understand the basics of Synopsys VCS.

2 Introduction

The Verilog HDL is one of the two most common Hardware Description Languages (HDL) used by integrated circuit (IC) designers (the other is VHDL). HDLs allow the design to be simulated earlier in the design cycle in order to correct errors or experiment with different architectures. Designs described in HDL are technology-independent, easy to debug, and are usually more readable than schematics, particularly for large circuits.

Verilog can be used to describe designs at four levels of abstraction:

1. Algorithmic level (much like C code with if-else, while, case and loop statements).

2. Register transfer level (RTL uses registers connected by Boolean equations).

3. Gate level (interconnected AND, NOR etc.).

4. Switch level (the switches are MOS transistors inside gates).

The language also defines constructs that can be used to control the input and output of simulation. Verilog is also used as an input for logic synthesis programs which will generate a gate-level description (a netlist) for the circuit. The netlist can then be loaded on to a chip called FPGA (Field Programmable Gate Arrays), which is a generic chip on which any functionality can be programmed and re-programmed when required.

3   Questions

1. What is the difference between Structural and Behavioral programming style? Give an example for each style.

2. **Half Adder**
   Design a half adder comprised of just $NAND$ gates (points will be deducted for using an XOR. Give the Truth Table for the circuit. Write a structural Verilog program for the half adder.

3. Develop a testbench for the Half Adder that verifies the structural model. The testbench will have no ports. Exhaustively simulate the circuit and print the output demonstrating that the model is correct. Text output can be generated using the `$monitor` and `$display` tasks.

4. **Multiplexor**

Design a 2-to-1 MUX Circuit, that multiplexes two input signals to a single output based on the select signal.
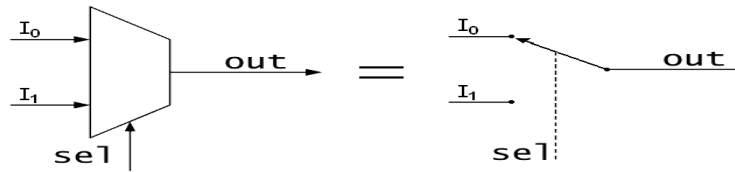

Fig. 1: Multiplexor

(a) Give the truth table for the circuit:

(b) Design the Optimal Circuit design using the basic gates (not to be submitted).

(c) Write a structural Verilog model for the 2-to-1 MUX:

(d) Develop a testbench for 2-to-1 MUX that verifies the structural model.