EC605: Computer Engineering Fundamentals Lab 2: ALU

Fall 2016

Combinational Logic and Verilog Simulation

Goals

- Introduction to Verilog design in Vivado.
- Testing via simulation in Vivado.
- More combinational circuits design.

Overview

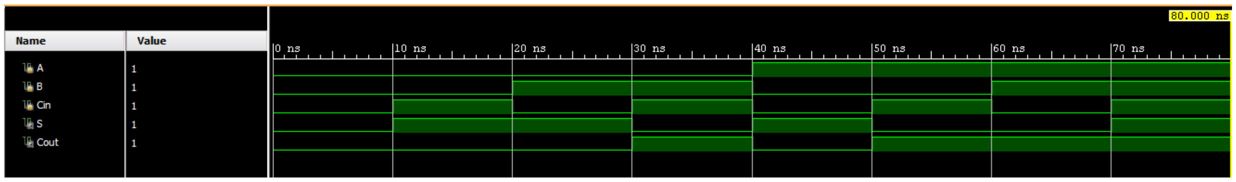
In this lab you will design structural-Verilog-based 8-bit ripple-carry and carry-select adders and a behavioral-Verilog-based 8-bit ALU. Your designs will be tested via simulation.

Tasks

Task 1: Gate-Level Full Adder

1. Build a 1-bit full adder from basic logic gates following the process that was described in class. Refer to the *Vivado Verilog Tutorial* to create a gate-level (structural) Verilog module for your full adder.

2. Use the provided Full_Adder_Test.v testbench to simulate your design, as described in the *Vivado Verilog Tutorial*. Ensure that your full adder works by testing the output for all input combinations. Your waveform should resemble the following:



Task 2: Ripple Carry Adder

1. In a new project, instantiate your full adder 8 times to implement a simple 8-bit ripple-carry adder. The ripple carry adder should be implemented in a separate file from the full adder. Your project should include both files. Note that instantiating the full adder is different from coping the code 8 times.

2. Modify the full adder testbench to test your ripple carry adder. In this case, testing all input combinations is not realistic. Instead, you will need to think of a representative input sample to test.

Task 3: Carry Select Adder

1. In a new project, design an 8-bit carry-select adder. You may instantiate your full-adder and ripple-carry adder as needed.

2. Modify the ripple carry testbench to test your carry select adder. Use the same input set that was used in the previous task, and verify that both adders give the same output.

Task 4: ALU

1. In a new project, design an 8-bit ALU using behavioral Verilog. Your ALU should receive two 8-bit inputs, A and B, along with a 4-bit opcode, OPCODE, and generate an 8-bit output, Y, and four one-bit flags, N, Z, C and V. The ALU functionality is defined as follows:

Opcode	Operation
1	AND
2	OR
3	NOT
4	XOR
5	Shift left
6	Arithmetic shift right (A >>> B)
7	Logic shift right
8	Add
9	Subtract

The flags are described as follows:

Flag	Operation
N - Negative	N = 1 when Y < 0
Z - Zero	Z = 1 when Y == 0
C – Carry	C =1 when A+B has a carry-out
V – Overflow	V =1 when A+B has an overflow

2. Create a new testbench and verify functionality of all ALU inputs.

3. Connect the ALU to the provided top module *Top_module.v* and constraints file *Lab2_Task4_ALU_Constrains.xdc*.

4. For the final set-up of this lab, your board should have the following functionality:
   - The ALU output should be displayed on the board's seven-segment display.
   - Each ALU flag should be connected to an individual LED.
   - The switches will be used to input values for A, B, ~~C,~~ and OPCODE.

5. The provided top module and constraints file contain the following as a guide.

- The constraints file has all of the connections between the top module and the FPGA board. This includes clk, rst, push buttons, switches, LEDs and seven segment.
- Two module headers are provided – these are optional, and you may decide if to use them:
    - **i.** Binary to Decimal converter: input an 8-bit binary number and output a 3 digit decimal number.
    - ii. Seven Segment Display: input a decimal digit and output 7 bits for each of the seven segments.

Hint: Use the Vivado Block Design Tutorial (or the reference manual information found online), as a guide to using the seven segment display.

Deliverables

- Submit your Verilog code and testbench for each task. Also submit a pdf with a waveform and a description of the tests done for each task.
- Sign-up to demo your design on the Xilinx FPGA board to a TA. Come prepared to answer questions on your design.