EC605: Computer Engineering Fundamentals Fall 2017

Lab 2: ALU

Combinational Logic and Verilog Simulation

Goals

- Continuation of learning Verilog design in Vivado.
- Creating behavioral simulations and implementing on the board for testing.
- Using modular thinking skills to create a multi-component simulation.

Overview

In this lab, you are expected to design a structural-Verilog-based 8-bit ripple-carry and carry-select adders and a behavioral-Verilog-based 8-bit ALU. For the structural-Verilog, you will be using simple gates that you have encountered in the Tutorial as well as Lab 1. For behavioral-Verilog, you will be introduced to new syntax in the Verilog language. Additionally, you will be introduced to thinking about Verilog programming using a modular method. You will be ex expected to test all of your designs via behavioral and on-the board simulation.

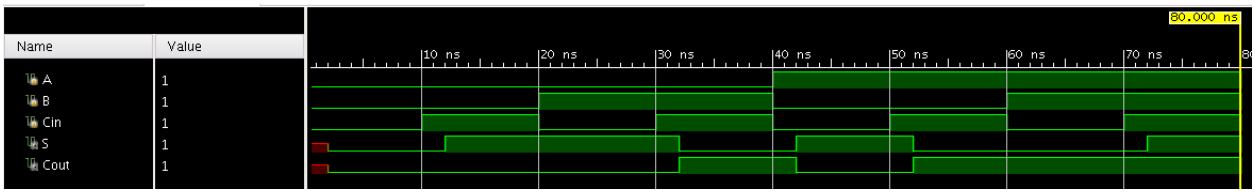
Tasks

Task 1: Gate-Level Full Adder

1. Create a new project.

2. 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.

3. Create a 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:



4. In Vivado, in the Flow Navigator tab, Click into Synthesis -> Synthesized Design -> Schematic. Let your schematic for the Gate-Level Full Adder load, take a screenshot and save it to the PDF you are submitting.

Task 2: Ripple Carry Adder

1. Create a new project.

2. Add a new source file as your top module. Add an existing source file (from Task 1) to instantiate your full adder 8 times to implement a simple 8-bit ripple-carry adder. Note that instantiating the full adder is different from coping the code 8 times.

3. 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.

4. Click into Synthesis -> Synthesized Design -> Schematic. Take a screenshot of the Ripple Carry Adder schematic and save it to the PDF you are submitting.

Task 3: Carry Select Adder

1. Create a new project.

2. Create and add a new source file to your new project (Carry_Select.v). This time, you want to instantiate your Full Adder and/or your Ripple Carry Adder as needed.

3. 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.

4. Click into Synthesis -> Synthesized Design -> Schematic. Take a screenshot of the Ripple Carry Adder schematic and save it to the PDF you are submitting.

Task 4: ALU

The overall goal of this task is for you to design a 4-bit ALU using Behavioral Verilog to implement on the board. The board should show the calculated answer in **hexadecimal** on the seven-segment display.

1. In a new project, design an 4-bit ALU using behavioral Verilog. Your ALU should receive two 4-bit inputs, A and B, along with a 4-bit opcode, OPCODE, and generate a 4-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. Hand-draw a block diagram of this entire module with the inputs and outputs as well as the wires (buses) you will be using. Take a picture or scan it and submit it as part of your PDF.

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

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 in **hex**.
   - Each ALU flag should be connected to an individual LED.
   - The switches will be used to input values for A, B, and OPCODE.
     4-bit input A, B, Opcode
   - A reset button to set display to 0 when pressed.

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

Deliverables

1. Submit your Verilog code and testbench for each task.
2. Submit a pdf with a waveform and a description of the tests done for each task. The waveforms and descriptions that should be submitted **for each task** are as follows:
   - Verilog file (.v)
   - Testbench (.v)
   - Waveform (ON PDF)
   - Vivado Schematic (ON PDF)

Sign-up to demo your design on the Xilinx FPGA board to a TA. Come prepared to answer questions on your design.