EC605: Computer Engineering Fundamentals - Fall 2024

Lab 2: ALU

Combinational Logic and Verilog Simulation

Goals

- Introduction to structural and behavioral Verilog using Vivado.
- Using hierarchical design to build more complex modules by instantiating simpler ones.
- Create behavioral simulations to test designs and implement them on the FPGA board.

Overview

In this lab, first make a full adder using structural Verilog (gate-level design). Then, design an 8-bit ripple carry adder using a modular approach by instantiating the full adder. Next, design a 4-bit ALU using behavioral Verilog (define outputs as operations on input signals). Test all modules with the behavioral simulation, and implement the ALU on the FPGA board.

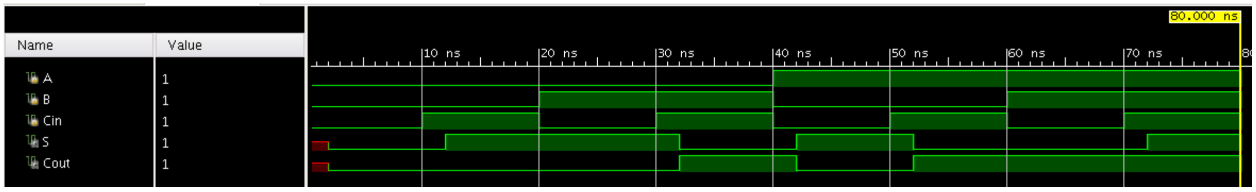
Tasks

Task 1: Gate-Level Full Adder

1. Create a new project.

2. Build a 1-bit full adder using basic logic gates. Refer to the lecture notes or the *Vivado Verilog Tutorial* for the correct gate design and the syntax for structural Verilog.

3. Create a 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. Refer to the lecture notes for the correct truth table. Your waveform should resemble the following:



Task 2: Ripple Carry Adder

1. Create a new design source for the 8-bit ripple carry adder. Instantiate your full adder 8 times to implement the ripple-carry adder. Note that instantiating the full adder is different from copying the code 8 times. See the lecture notes for an example of instantiation of modules.

2. Create a new testbench for the ripple carry adder. In this case, testing all input combinations is not realistic. Instead, you will need to think of a representative set of inputs to test.

Task 3: ALU

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

1. In a new design source, design a 4-bit ALU using behavioral Verilog. See the lecture notes for behavioral Verilog syntax. Your ALU should receive two 4-bit inputs, A and B, along with a 3-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	
000	AND	A AND B
001	OR	A OR B
010	XOR	A XOR B
011	Arithmetic shift right	A >> B (arithmeitc)
100	Logical shift left	A << B (logical)
101	Subtract	A - B
110	Add	A + B
111	Set less than	(A<B)? 1: 0

The flags are described as follows:

Flag	Operation
N - Negative	N = 1 when Y < 0
Z - Zero	Z = 1 when Y == 0
C – Carry-Out	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 (not gate level) 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 the functionality of all ALU operations.

4. Design a top module and program the FPGA board with it to test your ALU module.
   A file Top_Module.v is provided on Blackboard with the module skeleton needed to program the FPGA. To implement your ALU you should add to and modify the file as needed, instantiating your ALU module, connecting the inputs to the switch variables and the outputs to LEDs (flags) and the seven segment display (ALU output). You will need to specify which cathodes should be lit on the seven segment display for every possible hex outcome of the ALU. Remember also to create a constraints file and run synthesis before programming the board as done in *Vivado Verilog Tutorial*.

5. For the final demo 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 (11 switches in total).
   - A reset button to set the display to 0 when pressed. (You choose from buttons BTNL-BTNR on the board).

Hint: Use the *Vivado Verilog Tutorial* (or the reference manual information found online), as a guide to using the seven-segment display and remember, clock is connected to pin W5.

Deliverables

1. Submit your Verilog code and testbench for each task on Blackboard:

   - **Full Adder:** code and testbench .v files
   - **Ripple Carry Adder:** code and testbench .v files
   - **ALU:** code and testbench .v files

2. Submit a PDF including a waveform and a description of the tests done for each task, as well as the ALU schematic, on Blackboard:

   - **Full Adder:** waveform and description
   - **Ripple Carry Adder:** waveform and description
   - **ALU:** waveform and description, hand-drawn schematic

3. Program the FPGA board to demo to the TAs– for part 4 (ALU) only.

Sign-up to demo your design on the FPGA board to a TA (sign-ups on Blackboard). Come prepared to show your work and answer questions about your design.