ECEN2350 Digital Logic - Project 1

Due Friday March 5th (Midnight) - 30 Points [+3pts EC]

Overview

For this project you will become familiar with verilog utilizing many different verilog keywords. You will design a mini-CPU that can perform some very basic arithmetic, logical and comparison operations. There is also an extra credit Magic mode. You will utilize input from the switches and buttons, and output to the LEDs and the 7-Segment Displays. Your CPU needs to support a 4-bit word size (4-bit math).

You must use the DE10-LITE board (Available in the EE Store) with the MAX 10 10M50DAF484C7G Device. Groups of 1-2 students, with **1 Report / group.** Grading will be done on the CCC Rubric (See D2L).

No homework will be given for the second week of the project so students will focus on the project design.

Outcomes:

- Interface a design with Input and Output systems such as LEDs, 7-Segment Displays, buttons, and Switches
- Design a project using Verilog Hardware Description Language with behavioral specifications, using both procedural and continuous assignment
- Gain experience using an FPGA Development kit with the Quartus environment.

Verilog Module Requirements

A description of the modules you need to design are below:

- Project1_top.v
    - This should take in the 2-button keys and 10 switches.
    - This should output to 10 LEDs, and 14 7-segment bits (2x 7-seven segment displays).
    - The buttons KEY0 and KEY1 should determine specific modes
        - Arithmetic, Logical, Comparison, Magic
    - The switches SW8 and SW9 should determine specific operations within each mode
    - The buttons must use procedural assignment on only the **rising edge** button presses. This should will require the register keyword for the outputs.
    - This module should instantiate all necessary modules listed below. However, it will need to **control** which CPU operation is currently accessing the 7-segment displays (**Data path**). Meaning, only one mode and one operation may be active at any time. (hint: Think Multiplexers)

- SevenSegment.v
    - This needs to take in 4 binary inputs and output 7 binary digits to control the seven segment display
    - You must support all Hex characters for display
    - You must use **procedural assignment** with a **case statement**.
- Arithmetic.v
    - These are basic arithmetic operations. If you produce a carry/remainder, you should turn on the LED9.
        - Add (x[3:0], y[3:0])
            - Output the result of x+y and the carry if it exists
            - You must write this using a n-bit Ripple Carry adder Implementation with **parameters**, **genvar**, **for** and the **generate** keyword.
        - Subtract (x[3:0], y[3:0])
            - Output the result of x-y and the carry if it exists
            - You must write this using a n-bit Full Subtractor Implementation with **parameters, For Loop** and **integer** statements.
        - Multiple-by-2 (z[7:0])
            - Output the result of z*2 and the carry if it exists
        - Divide-by-2 (z[7:0])
            - Output the result of z/2 and the remainder if it exists
    - Test output should be provided with x = 0xA, y = 0x7. z = 0xA7
- Logical.v
    - All code in this module needs to use **continuous assignment**.
        - AND (x[3:0], y[3:0])
            - Bitwise Logical AND of x and y
        - OR (x[3:0], y[3:0])
            - Bitwise Logic OR of x and y
        - EXOR (x[3:0], y[3:0])
            - Bitwise logical EXOR of x and y
        - NOT (z[7:0])
            - Bitwise logical not of z (all 8 switch inputs)
    - Test output should be provided with x = 0xA, y = 0x7. z = 0xA7
- Comparison.v
    - For each of the following operations you should output a simple 1 or 0 if X <operation> Y is True or False. E.g for x=2,y=3, x>y should output a 0.
        - EQUAL(x[3:0], y[3:0])
            - 1 if they are equal
            - 0 if they are not equal
        - GREATER(x[3:0], y[3:0])
            - 1 if x > y
            - 0 if y >= x
        - LESS-THAN(x[3:0], y[3:0])

                - 1 if x < y
                - 0 if y <= x
            - MAX (x[3:0], y[3:0])
                - X value if largest
                - Y value if largest
        - Test output should be provided with x = 0x8, y = 0x7.
- Magic.v (Extra Credit - 3pts)
    - Utilize a clock and have the nightrider LED pattern output on the LEDs
        - https://youtu.be/aNnM0Zy2qYE?t=38s
- Multiplexer.v
    - This needs to take in a all of the outputs from the cpu operation modules and the mode buttons, to output to the 7-segment displays.
    - You should need 2 multiplexer instantiations
    - These should be **~like** a 4-1 multiplexer implementation, where each 1 input of data is actually lots bits (the output of the operation modules).

Deliverables

Procedures: You need to include the following items in the report.

- Start with designing a block diagram for the project before you go into coding. This is a requirement for the report. This should show all your modules connect (basically a picture of what the Project1_top.v file is doing.
- You need to determine a bitwise encoding for the modes and operations given your switches and buttons. Provide a table that describes your encoding scheme for that.
- You need to take pictures for each of the Modes and operations showing that the provided encoding matches the intended output for the given test outputs.
- (Extra Credit) A short demo to the instructor team of the knight rider circuit in action

Report: A report with the following information.

- **Cover Page**: Team Members names, class, project
- **Introduction**: 1-paragraph introduction on what the project is including a block diagram of your design.
- **Lab procedure/Results**: Answer the questions in the procedure section. Explain how your circuit function. You should provide
- **Conclusion:** 1-paragraph conclusive statement on the project. What was the most difficult part of this project? If something did not work, what is the reason and how would you correct it? What would you do different next time?
- **Appendix**: Original Source of your nicely formatted verilog *.v files.

Resources

- Quartus 16.1 Prime Software Lite Edition (Free): https://www.altera.com/downloads/download-center.html
- Be sure to install Modelsim and the components for the DE10-LIT MAX10 FPGA chipset.
- Terasic Informaiton: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1021