CS224 - Fall 2016 - Lab #5 (Version 1: Nov. 11, 4:30)

Concepts of the Single-cycle Datapath/ Review of HDL and FPGA

Dates: Section 3, Wednesday, 16 November, 08:40-12:30
Section 1, Wednesday, 16 November 13:40-17:30
Section 4, Thursday, 17 November 13:40-17:30
Section 2, Friday, 18 November 13:40-17:30

Note: Please bring and drop your preliminary work into the box provided in front of the lab by 5:00 pm Tuesday November 15. Note that in the lab document we will use Verilog but you are free to use System Verilog if you like.

Submit your code to the Unilica > Assignment specific for your section. You will upload **one file** with *name_surname_HDL.txt* format. Be sure that the file contains exactly and only the codes which are specifically detailed below. *Even if you didn't finish, or didn't get the HDL codes working, you must submit your code to the Unilica Assignment for similarity checking.* Your codes will be compared against all the other codes in the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism).

You have to show your work to your TA and upload it before by 11:30 in the morning lab and by 4:30 in the afternoon lab. Note that you cannot wait for the last moment to do this. Make sure that you have shown your work to the Lab TA before uploading. If you wait for the last moment and show your work after the deadline time 30 points will be taken off. If you upload before showing your work to the TA you will receive 0 (zero) for the lab work.

You also can find helpful guide about this lab at the link bellow:

Survival guides for FPGA and Verilog labs

Part 1. Preliminary Work Related to Single-Cycle Datapath (30 points)

Question 1)

a) A single cycle datapath is constructed from different units. Mention these units (e.g., Instruction Memory etc.)in the order of appearance in the datapath, explain what each part is doing?

b) What are inputs and outputs of each module. How many bits are them?

c) Which control signals you need for the datapath? What do these signals control?

d) Write a module header for each unit with its inputs and outputs. It is enough to write each module definition for example you can write like this:

Module example(input a, output b)

You should write list of all input and outputs for your defined module.

e) Write a top level combinational module in Verilog (as stated above System Verilog is acceptable for all such cases) to connect these module to each other for building single cycle datapath.

f) On the single cycle datapath specify your modules on it (location of modules on the datapath). In this part you may take the datapath picture from the slides of the textbook and write the module names and its inputs and outputs on that figure by hand.

Question 2)

Part of a Verilog code of a decoder is written below:

```
////////////////////////////////////////////////////////

reg [8:0] controls;

 assign {regwrite, regdst, alusrc, branch, memwrite, memtoreg, aluop, jump} = controls;

case(op)

      6'b000000: controls <= 9'b110000100;

      6'b100011: controls <= 9'b101001000;

      6'b001000: controls <= 9'b101000000;

      default:   controls <= 9'bxxxxxxxxx;

////////////////////////////////////////////////////////
```

As you can see this part of the code wants to find control signals of three operations. All of control signals are one bit except “aluop” which is 2 bits.

a) What the value of “op” represents?

b) Specify type of each operation. How can you do it?

c) What does this line of code shows: “6'b100011: controls <= 9'b101001000;”

Why some bits of “controls” are 1 and others are 0? Explain its reason.

d) What is the meaning of final line of the code (default case)?

Part 2. Instruction Decoding
(40 points)

a) The aim of the *main_decoder* module is to generate appropriate control signals of the single cycle datapath for **R-types**, **sw**, **lw**, **j**, **beq**, **addi** instructions. So, its input is an opcode of an instruction and its outputs are control signals. This module is given to you in "*main_decoder.txt*" file. It only has three instructions and their resulted final control signals. You should add three other instructions not implemented in the given code and generate their control signals and complete the module.

b) Then, you need to make a new project in Xillinx with a meaningful name and add your completed Verilog module to it. Compile *main_decoder* file in Xillinx and simulate it using a testbench to test all of instructions in the decoder module. You should be able to analyze input and output of your simulation result and understand all steps of your work.

   Implement this module on BASYS (and Beti board if needed) and see resulted control signals. For output control signals use LEDs and use switches for getting input (instruction) from user.

Part 3. (30 points)

Pushbutton Switch Exercise

The *debouncer* module is given to you with pulse_controler.txt file.

a) Understand the code to be able to answer questions asked by your TA. You should figure out what it is doing and what each part of the code is related to.

b) Write a testbench for this module and show how it is working. You must show how it is working with testing sufficient values and show how it ignores the changes for the debounce period. Please take this into account that code has been already written. You have to understand it and show it is working in simulation correctly. Only simulation without understanding of the module does not have any point.

   Implement this module on BASYS (and Beti board if needed) and see resulted clk_pulse. For CLK you should use push button of Beti board.

*** Important note ***

For all of parts, your understanding of all of your works, ability to explain it and answer to related TA questions are necessary to get the grade.

Part 4. Submit your code for MOSS similarity testing

1. Your test bench file used in part 2 and 3
2. User constraint file for mapping pins(.ucf file) for part 2 and part 3

Part 5. Cleanup

1) Erase all the files that you created.
2) When applicable put back all the hardware, boards, wires, tools, etc where they came from.
3) Clean up your lab desk, to leave it completely clean and ready for the next group who will come