EECE 3270: INTRO TO MICRO
SPRING 2020

Instructor: Dr. B. I. Morshed, EECE, The University of Memphis

PROJECT GUIDELINE

[Subject to amendments]

- Project must be performed in a group setting (as instructed in the lab). Lab groups will be formed by the Instructor.
- The quality, depth and attention to details of the project deliverables (as outlined below) will determine the overall grade of each student.
- Individual marks might be different for students of the same project group, as individual's grading will be based on both overall group performance and individual's performance.
- **Project hardware implementation can ONLY be completed with the DE0 boards available in ET227 lab.**
- **Each group will be checked out 1 Altera DE0 board from the lab, which must be returned immediately after the final project demonstration.**

- **Project Deliverables** and grading scheme **(total marks: 10)**:
    1. **Project Presentation:** Presented in class using power-point (or similar) slides. Time allowed: 5 mins for each group. Each individual must present his/her portion of the work – **5 marks**
    2. **Project Demonstration:** After all presentations are complete, each group needs to setup the hardware for demonstration **- 1 marks**
    3. **Project Report:** Each group must submit 1 report (assimilated from all group members). The report must contain a clear demarcation of each individual's activities within the group. The report is due on the day of the presentation/demonstration. Report must be submitted via eCourseWare by any one of the group members – **4 marks**

Some examples project presentation videos from previous years:
https://www.youtube.com/watch?v=bA09yZgeABY
https://youtu.be/ZMd-9GuqinA
https://www.youtube.com/watch?v=CUZ_xAQJOww
https://www.youtube.com/watch?v=sshWaxvXkNc
https://www.youtube.com/watch?v=LC4GHJlFbzc
https://www.youtube.com/watch?v=2y31n5NR9Yg

Objectives:

- To enable students to thoroughly understand and appreciate the internal functional blocks of a typical microprocessor, and the microarchitecture of a MIPS single-cycle microprocessor.
- To enable students to understand execution of instructions in a MIPS microprocessor at assembly / machine language level of programming.

- To be able to formulate, design and analyze engineering problems of a MIPS microprocessor using ALU, registers, memory, control unit, and other functional blocks with an FPGA development board.
- To use hardware and software for a constrained engineering design.
- To effectively communicate an engineering outcome to a peer audience.

Project Overview:

This course project objective is to develop an HDL (either VHDL, Verilog, or SystemVerilog) code for a MIPS microprocessor design. The processor architecture is based on the textbook: the MIPS single-cycle processor. The project requires you to get very familiar with the single-cycle implementation of the MIPS processor described in Section 7.3-7.6 of your textbook, *Digital Design and Computer Architecture*. You should follow the codes given in these sections of the textbook, which includes (most of) the necessary modules along with some testbenches. The project members must encode these modules using Altera Quartus II software, and independently verify functionality of each module using testbenches, then combine all of the required modules to design the complete microprocessor. After complete system is functionally verified using testbench, the microprocessor design should be downloaded to the Altera DE0-nano (or Altera DE0) FPGA board for final evaluation. Memory needs to be programed with the required instructions preloaded in memory. Utilize a push Key as the clock for the microprocessor and another push Key for RESET, so that you can assess and evaluate the output after each instruction. 7-segment LEDs can be used to (partially) display the output bits (e.g. ALU outputs, or register contents) of the microprocessor for each instruction operation.

Brief Description of Fundamentals:

Model of the single-cycle MIPS processor by dividing the design into two major units: the control-path and the data-path. Each unit is constructed from various functional blocks. For example, the data-path contains the 32-bit ALU, the register file, the sign extension logic, and five multiplexers to choose appropriate operands (consult chapter 6 and 7 of the textbook).

Within the given code in the textbook, look at the MIPS module, which instantiates two sub-modules: **controller** and **datapath**. Then, take a look at the controller module and its submodules. It contains two sub-modules: **maindec** and **aludec**. The maindec module produces all control signals except those for the ALU. The **aludec** module produces the control signal, **alucontrol[2:0]**, for selction control of the ALU. Make sure you thoroughly understand the controller module, as it is very important for all submodules. Correlate signal names in the HDL code with the wires on the schematic. The datapath has quite a few submodules. Make sure you understand why each submodule is included, and where each is located on the MIPS single-cycle processor architecture.

The highest-level module, **top**, includes the instruction and data memories as well as the processors. Synthesize the highest-level module, **top**. Notice that the only necessary inputs to the highest-level module are **clk** and **reset**. The other signals are there for verification purposes only.

View the synthesis report. There should be no errors or warnings (you can confirm this by looking at the errors tab at the bottom of the screen). View the RTL schematic to understands blocks in each module. If it does not look as you expected, fix the errors and resynthesize.

A good way to verify functionality is to begin by predicting what should happen on each cycle when running the program. View the sources for Behavioral Simulation, and look at the testbench module.

Each of the memories is a *64-word × 32-bit* array. The instruction memory needs to contain some initial values representing the program. To use this test code, it must be loaded into the MIPS instruction memory.

It would have been nice to be able to write Verilog code for a memory with some initial values (the test program), but the software does not support this very well yet. So, instead, you can generate ROM to produce a memory with initial values. The instruction memory, **imem**, will be constructed as a ROM that will hold the program (the instructions) to execute. Note that this is a challenging task, so ensure you leave sufficient time to complete this part of the project.

Make sure the module and the software program works in simulators first (Quartus II and ModelSim) before programming (downloading) onto the board. To test the processor, you will simulate running .asm (assembly file).

Clearly demonstrate the functional verification of the followings in the project presentation and the project report (in order of indicated priority sequence):

1. This MIPS single-cycle processor executing the following instructions: ***A + B, A - B, A and B, A or B, A and ~B,*** *&* ***A or ~B***. You must demonstrate these by presenting the timing simulation results (by writing required testbenches that functionally verifies each instruction).
2. **Required minimum instructions:** If you successfully complete (1), then implement the followings: ***slt, lw, sw, beq, addi,*** and ***j***. Note that you will need to add the required hardware in your microprocessor design for successful implementation of all of these instructions. Write testbenches to functionally verify each of these instructions (through timing simulation).
3. Determine the resulted RTL diagram of the microprocessor design. Identify various major elements of the design in the RTL diagram. Determine if the RTL matches with expectations, or else determine the reason for discrepancy.
4. After that, demonstrate the functionality of all of the above instructions with hardware programming of the designed microprocessor by downloading the entire design code to the Altera DE0 FPGA development board hardware platform, and executing instructions that encompass all of the encoded instructions (by writing sample instructions and observing the ALU outputs/register contents that validates as per expectation). Ensure that all of the above instructions are verified at least once in your sample instruction sequence.
5. ***Bonus 1:*** *(up to 10% of the project mark)* Implement **ori** & **bne** instructions by modification of your ALU and other hardware based on the textbook after implementing the above listed instructions for bonus marks of up to 10% (provided that the required list of instructions are properly working as well). You must provide evidences of all instructions to be working (including the required instructions after modification).
6. ***Bonus 2:*** *(up to 10% of the project mark*) If your group create a video that demonstrates functionalities of the downloaded microprocessor code, and properly submit it to instructor with permissions from all group members to upload it in the course related YouTube channel (for future year students), your group will be eligible for another 10% (of the project mark) as bonus. The video file must be in mp4, mov, or any other format that can be directly uploaded to YouTube. The entire group will get the bonus mark when the instructor is able to upload it through his research lab YouTube channel (esarplab umemphis) for public access.

Items that must be included in the project report (ONLY soft copy, upload to eCourseware, by any ONE member of the project team):

1. You must show the final functional verification waveforms (timing diagrams) for all implemented instructions using this order: **clk, reset, pc, instr, aluout, writedata, memwrite,** and **readdata**. All the values need to be output in hexadecimal format, and must be readable in the report (high resolution, or sufficiently zoomed). For example, ensure that the waveforms are zoomed enough that the grader can read the bus values, as well as see the appropriate changes as expected. Unreadable waveforms might not receive full score. You can use multiple pages for each graph if necessary.
2. Include important aspects of the design, such as RTL diagram of critical modules, final HDL codes along with testbenches, etc. (All codes including in the report must be properly annotated, commented, and documented for ease of understanding).
3. Include any other simulation waveforms and functional verification results that are important for your design (e.g., pinout).
4. Include marked up versions of the datapath schematic (RTL). Also provide the decoder table and explain each entry. Provide RTL of the control unit and explain the control unit hardware.
5. Demonstrated in the report that input and output waveforms in the timing simulation to match with expectations for proper functionality.
6. Provide downloading process, snapshots, and description of the final hardware that clearly demonstrate functional verification on hardware. Describe the pin diagram and the connection for switches and 7-segment LEDs.

Note: At the beginning of the report, a clear table must be provided that explains what each team member contributed to for the entire project work.