CS224 – Fall 2017 - Lab #5

Implementing the MIPS Processor with Pipelined Microarchitecture

Dates:

Section 1:	Wed, Dec. 6th	13:40-17:30 in EA-Z04
Section 2:	Fri, Dec. 8th	13:40-17:30 in EA-Z04
Section 3:	Wed, Dec. 6th	08:40-12:30 in EA-Z04
Section 4:	Thu, Dec. 7th	13:40-17:30 in EA-Z04

Purpose: In this lab you will convert the single-cycle MIPS processor into a pipelined processor, using the digital design engineering tools (System Verilog HDL, Xilinx Vivado and FPGA, Digilent BASYS3 development board) to implement and test the pipelined MIPS. To do this, you will need to add pipeline registers, forwarding MUXes, a hazard unit, etc to your datapath, and of course make the control pipelined as well. To implement these changes, converting the single-cycle machine into a pipelined version will require you to modify some System Verilog modules in the HDL model of the processor, then synthesize them and demonstrate on the BASYS3 board.

Summary

Part 1 (50 points): Preliminary Report/Preliminary Design Report: Verilog model for Pipelined MIPS processor handlingOriginal10 + New instruction (Due date of this part is the same for all).

Part 2 (50 points): Simulation of the MIPS-lite processor and Implementation and Testing of new instructions.

DUE DATE OF PART 1: SAME FOR ALL SECTIONS Dear students please bring and drop your preliminary work into the box provided in front of the lab before 3:59 pm on Tuesday December 5h. No late submission!

LAB WORK SUBMISSION TIMING: You have to show your lab work to your TA by **12:15** in the morning lab and by **17:15** in the afternoon lab. Note that you cannot wait for the last moment to do this. If you wait for the last moment and show your work after the deadline time 20 points will be taken off.

If we suspect that there is cheating we will send the work with the names of the students to the university disciplinary committee.

Part 1. Preliminary Work / Preliminary Design Report (50 points)

Make multiple printouts of the datapath and 2 tables in the "MIPS datapath and control tables.doc" file. Now you have the datapath and the control tables (for main decoder and ALU decoder) to add new instructions and pipelining to the single-cycle MIPS processor (This file is found in Unilica in /CS224/Resources/MIPS). Be sure to make a copy of the report, to use during the lab. Your PDR should contain the following 7 items, one per page:

a) Cover page, with university name, department name, and course name and number at the top, "Preliminary Design Report", Lab # (e.g 5), Section #, and your name and ID# in the middle, and the date of your lab at the bottom.

b) **[5 points]** MIPS processor datapath drawing for the 10-instruction "mini-MIPS" machine that does the Original10 instructions, modified to include the new (or modified) hardware and data paths needed to make it pipelined. The branch decision should be made in the Execute stage. The base datapath should be in black, with changes marked in red.

c) **[5 points]** The list of all hazards that can occur in your pipeline. For each hazard, give its type (data or control), its specific name ("compute-use" "load-use", "load-store" "J-type jump", "branch" etc), the pipeline stages that are affected, the solution (forwarding, stalling, flushing, combination of these), and explanation of what, when, how.

d) **[5 points]** The logic equations for each signal output by the hazard unit, as a function of the input signals that come to the hazard unit. This hazard unit should handle all the data and control hazards that can occur in your pipeline (listed in c) so that your pipelined processor computes correctly.

e) **[5 points]** Make a list of the System Verilog modules (not files, modules) that will need changes in order to make the MIPS processor a pipelined microarchitecture. Any new modules, such as you will need for the hazard unit, should also be listed. For each module, indicate if it is new, or changed.

f) **[20 points]** For each module in the part e) list, give the System Verilog code.

g) **[10 points]** Write some test programs, in MIPS assembly, that will show whether the pipelined processor is working or not. Each of your test programs should be designed to catch problems, if there are any, in the execution of MIPS instructions in your pipelined machine. You can use the student-written assembler tool available online to help you quickly implement your test programs. Remember that the goal of testing is to verify that all the instructions are fully working, and that all the instructions are working even in the presence of hazards. [Hint: write a no-hazard test program, to verify that the pipeline works correctly. Then write a program that has one kind of hazard, then another, then another, etc until all the hazards have been tested.]

Table of instructions to implement-- by section

Section	MIPS instructions
Sec 1	Base: Original10 New: “subi”
Sec 2	Base: Original10 New: “sw+”
Sec 3	Base: Original10 New: lui
Sec 4	Base: Original10 New: jalr

The Original10 instructions in “MIPS-lite” are add, sub, and, or, slt, lw, sw, beq, addi and j. If you want to challenge yourself (and impress the TA !), you may optionally add other instructions from Lab 4.

Part 2: Simulation and Implementation

Implementation (25 points)

To implement the modified processor, you will need to assemble the modified System Verilog modules that model the changes you made to implement the pipelined microarchitecture. You should also consider the following: to slow down the execution to an observable rate, the clock signal should be hand-pushed, to be under user control. One clock pulse per push and release means that instructions advance one stage in the pipeline. Once the pipeline is full, it means that each push will cause one instruction to finish unless a flush or stall caused nothing to complete that cycle. Similarly the reset signal should be under hand-pushed user control. So these inputs need to come from push buttons, and to be debounced and synchronized. The memwrite and regwrite outputs (along with any other control signals that you want to bring out for viewing) can go to LEDs, but the low-order bits of writedata (which is RF[rt]) and of dataaddr (which is the ALU result) should go to the 7-segment display, in order to be viewed in a human-understandable way. [Consider why it isn’t necessary to see all 32 bits of these busses, just the low-order bits are enough.] So a new top-level System Verilog module is needed, containing top.v as well as 2 instantiations of pulse_controller, and 1 instantiation of display_controller, and some hand-pushed signals coming from push buttons, and some anode and cathode outputs going to the 7-segment display unit on the BASYS3 board, and some signals going to LEDs.

For each module that you wrote or modified, you will want to test it in simulation, to make sure it works functionally.When you have integrated all the System Verilog modules together and your whole pipelined MIPS is working in simulation with the test programs you wrote, call the TA and show it for grade. To get full points from this part, you must know and understand everything about what you have done. (This part is worth 25 points.)

Hardware Testing (25 points)

Make the constraint file that maps the inputs and outputs of your top-level System Verilog model to the inputs (100 Mhz clock and pushbutton switches) and outputs (AN and CA signals to the 7-segment display, and memwrite (plus others?) going to a LED) of the BASYS3 board and its FPGA. Now create a New Project, and implement it on the BASYS3 board, and test it. When your pipelined instructions are working correctly in hardware, call the TA and show it for grade. Note: the TA will ask questions to you, in a single 25-point demo and Oral Quiz, to determine how much of the 25 points for this part of Part 2 (implementation) is deserved, based on your demo, your knowlege and ability to explain it and the System Verilog code and the reasons behind it, and what would happen if certain changes were made to it. To get full points from the Oral Quiz, you must know and understand everything about what you have done. (This part is worth 25 points.)

Part 3. Submit your code for MOSS similarity testing

In Part 2, you modified several System Verilog modules for parts of the MIPS that needed to change in order to be a pipelined implemention. You also created some new modules. Now it is time to combine all the new and modified Verilog codes into a file called YourName_YourSurname_Lab5.txt . You will then upload this file to the Unilica > CS224 > Assignment for your section. While the TA or Tutor is watching, you will upload this file. Be sure that this .txt file contains exactly and only the codes which are specifically detailed above (new or modified). Check the specifications! *Even if you didn't completely finish, or get the Verilog codes working, you must submit the YourName_YourSurname_Lab5.txt file to the Unilica Assignment for similarity checking.* If you don't submit your code, your grade for the lab will be 0 (see Lab Policies section, below NOTES.) 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 plaigarism). So be sure that you only submit code that you actually wrote yourself ! All students must upload their code to Unilica > Assignment while the TA or Tutor is watching, and before the deadline. NOTE: you are allowed to upload only ONE file to Unilica, so be sure it contains exactly and only the codes required.

Part 4. Cleanup

1) After saving any files that you might want to have in the future to your own storage device, erase all the files you created from the computer in the lab.
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.

LAB POLICIES

1. You can do the lab only in your section. Missing your section time and doing in another day is not allowed.
2. Students will earn their own individual lab grade. The questions asked by the TA will have an effect on your individual lab score.
3. Lab score will be reduced to 0 if the code is not submitted for similarity testing, or if it is plagiarized. MOSS-testing will be done, to determine similarity rates. Trivial changes to code will not hide plagiarism from

MOSS—the algorithm is quite sophisticated and powerful. Please also note that obviously you should not use any program available on the web, or in a book, etc. since MOSS will find it. The use of the ideas we discussed in the classroom is not a problem.

4. You must be in lab, working on the lab, from the time lab starts until your work is finished and you leave.
5. No cell phone usage during lab.
6. Internet usage is permitted only to lab-related technical sites.
7. For labs that involve hardware for design you will always use the same board provided to you by the lab engineer.