CS224 – Spring 2019 - Lab #5 (Version 2: April 8, 5:22 pm)

Implementing the MIPS Processor with Pipelined Microarchitecture

Dates: Section 1, Monday, 15 April, 13:40-17:30
Section 3, Tuesday, 16 April, 13:40-17:30
Section 2, Wednesday, 17 April, 13:40-17:30
Section 4, Thursday, 18 April, 13:40-17:30
Section 5, Friday, 19 April, 8:40-12:30
Section 6, Friday, 19 April, 13:40-17:30
Lab Location: EA-Z04

Purpose: In this lab you will implement and test a pipelined MIPS processor using the digital design engineering tools (System Verilog HDL, Xilinx Vivado and FPGA, Digilent BASYS3 development board). 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. You will be provided a skeleton System Verilog code for the Pipelined MIPS processor and fill the necessary parts to make it work. Then, you will synthesize them and demonstrate on the BASYS3 board.

Summary
Part 1 (60 points): Preliminary Report/Preliminary Design Report: Verilog model for Pipelined MIPS processor handling Original10 instruction (Due date of this part is the same for all).
Part 2 (40 points): Simulation of the MIPS-lite pipelined processor and testing on BASYS3 board.

DUE DATE OF PART 1: SAME FOR ALL SECTIONS:

1. Please bring and drop your printed preliminary work into the box(es) provided in front of TA office room number EA-407 by 13:30 on Monday April 15.
2. Please **upload your programs of Part 1 (PRELIMINARY WORK)** to the Unilica by 13:30 on Monday April 15. Use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.txt** Upload only the programs. Only a NOTEPAD FILE (txt file) is accepted. Your paper submission for preliminary work must match MOSS submission.

   We use MOSS testing only for code submissions. So, you are not supposed to submit the whole preliminary report online. You will ONLY submit the code online.

No late submission will be accepted.

DUE TIME OF PART 2—DIFFERENT FOR EACH SECTION:
You have to demonstrate your lab work to the TA for grade by **12:15** in the morning lab and by **17:15** in the afternoon lab. Your TAs may give further instructions on this. If you wait idly and show your work last minute, 20 points may be taken off from your grade.

At the conclusion of the demo for getting your grade, you will **upload your lab (only lab) program work** to the Unilica Assignment, for similarity testing by MOSS. See Part 3 below for details.

For further possible instructions etc. please follow the possible updates on the Unilica course web page. In the lab your TAs may also give further instructions.

Part 1. Preliminary Work / Preliminary Design Report (60 points)

At the end of this lab, you will have implemented the pipelined MIPS architecture that can be seen in the file that is provided as ***PipelineDatapath.PNG*** (Notice that there is no early branch prediction in this pipeline. Hence, the branch resolution is done in the Execute stage. Note also that hardware for jump instruction is not present; therefore, you do **NOT** need to implement it). Be sure to have a printout of the pipelined processor with you, to use during the lab. Your PDR should contain the following items:

a) You have to provide a neat presentation prepared by Word or a word processor with similar output quality such as Latex as you were asked in Lab 1. At the top of the paper on left provide the following information and staple all papers. In this part provide the program listings with proper identification (please make sure that this info is there for proper grading of your work, otherwise some points will be taken off).
CS224
Section No.: Enter your section no. here
Spring 2019
Lab No.
Your Full Name/Bilkent ID

b) **[13 points]** The list of all hazards that can occur in this 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.

c) **[8 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 b above) so that your pipelined processor computes correctly.

d) **[30 points]** You are given a skeleton System Verilog code for your pipelined MIPS processor in the file ***PipelinedMIPSProcessorToFillNew.txt***. The places in the code that needs to be modified are shown with comment blocks above them. Fill them and **highlight the changes you made** in the code **in your report**. You can use a different text highlight color (do this by hand after getting the printout) for this purpose. You do **NOT** need to follow the skeleton code point by point. If you think your design is better, you are welcome to try it in your code, as long as your version of the code works, too. Note that this is a design problem and there is no single solution to it but if you feel comfortable with the skeleton code then you can use it.

e) **[9 points]** Study the four example programs which respectively have no hazard, compute-use hazard, load-use hazard and branch-hazard. For hazardous programs explicitly show the places where hazards occur and the registers causing the hazard.

- No Hazard

`addi $t0, $zero, 7`	`8'h00: 32'h20080007;`
`addi $t1, $zero, 5`	`8'h04: 32'h20090005;`
`addi $t2, $zero, 0`	`8'h08: 32'h200a0000;`
`addi $t3, $t0, 15`	`8'h0c: 32'h210b000f;`
`add $t2, $t0, $t1`	`8'h10: 32'h01095020;`
`or $t2, $t0, $t1`	`8'h14: 32'h01095025;`
`and $t2, $t0, $t1`	`8'h18: 32'h01095024;`
`sub $t2, $t0, $t1`	`8'h1c: 32'h01095022;`
`slt $t2, $t0, $t1`	`8'h20: 32'h0109502a;`
`sw $t0, 2($t1)`	`8'h24: 32'had280002;`
`lw $t1, 0($t0)`	`8'h28: 32'h8d090000;`
`beq $t0, $zero, 1`	`8'h2c: 32'h1100fff5;`
`addi $t2, $zero, 10`	`8'h30: 32'h200a000a;`
`addi $t1, $zero, 12`	`8'h34: 32'h2009000c;`

- Compute-use hazard

`addi $t0, $zero, 5`	`8'h00: 32'h20080005;`
`addi $t1, $t0, 6`	`8'h04: 32'h21090006;`
`add $t2, $t1, $t0`	`8'h08: 32'h01285020;`

- Load-use hazard

addi $t0, $zero, 5	8'h00: 32'h20080005;
addi $t1, $zero, 6	8'h04: 32'h20090006;
addi $a0, $zero, 1	8'h08: 32'h20040001;
addi $a1, $zero, 2	8'h0c: 32'h20050002;
sw $t0, 0($t1)	8'h10: 32'had280000;
lw $t1, 1($t0)	8'h14: 32'h8d090001;
add $t2, $t1, $a0	8'h18: 32'h01245020;
sub $t2, $t1, $a1	8'h1c: 32'h01255022;

- Branch hazard

addi $t1, $zero, 2	8'h00: 32'h20090002;
beq $zero, $zero, 2	8'h04: 32'h10000002;
addi $t1, $zero, 5	8'h08: 32'h20090005;
addi $t1, $t1, 6	8'h0c: 32'h21290006;
addi $t1, $zero, 8	8'h10: 32'h20090008;
addi $a0, $zero, 0	8'h14: 32'h20040000;
addi $a1, $zero, 0	8'h18: 32'h20050000;
sw $t1, 0($zero)	8'h1c: 32'hac090000;

Part 2: Simulation and Hardware Testing

Simulation (20 points)

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.

Hardware Testing (20 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.

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 20-point demo and Oral Quiz, to determine how much of the 20 points for this part of Part 2 (implementation) is deserved, based on your demo, your knowledge 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.

Part 3. Submit your code for MOSS similarity testing

Combine all the new and modified Verilog codes into a file called **StudentID_FirstName_LastName_SecNo_LabNo_LAB.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* **StudentID_FirstName_LastName_SecNo_LabNo_LAB.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 plagiarism). 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.

Part 5. 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.