CS224 - Fall 2024 - Lab #5 (Version 1: November 18 2024; 7:32 pm)

Implementing the MIPS Processor with Pipelined Microarchitecture

Dates: (TAs - Tutor)

Section 1: Wed, 27 Nov, 13:30-17:20 in EA-Z04 (TA: Kadri, Soheil; Tutor: Deniz: 15:30-17:20)
Section 2: Thu, 28 Nov, 13:30-17:20 in EA-Z04 (TA: Onur, Soheil; Tutor: : Deniz: 13:30-15:20)
Section 3: Thu, 28 Nov, 8:30-12:20 in EA-Z04 (TA: Onur, Pouya; Tutor: Mert)
Section 4: Fri, 29 Nov, Fri 13:30-17:20 in EA-Z04 (TA: Berkan, Kadri; Tutor: Umut)

TA Full Name (email address: @bilkent.edu.tr)
Berkan Şahin (berkan.sahin)
M. Kadri Gofralılar (kadri.gofralilar)
Onur Yıldırım (o.yildirim)
Pouya Ghahramanian (ghahramanian)
Sepehr Bakhshi (sepehr.bakhshi) (Coord.)
Soheil Abadifard (soheil.abadifard)

Tutor Full Name (email address: @ug.bilkent.edu.tr)
Deniz Çatakoğlu (deniz.catakoglu)
Mert Emre Yamalı (emre.yamali)
Umut Başar Demir (basar.demir)

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

Preliminary Work: 50 points

1. **Part 1**: Preliminary Report/Preliminary Design Report: SystemVerilog model for pipeline registers and the hazard unit.

Lab Work: 50 points

2. **Part 2**: Simulation of the MIPS-lite pipelined processor and testing on BASYS3 board.

[REMINDER!] In this lab, we assume SystemVerilog knowledge, since you all took CS223 – Digital Design. If you are not familiar with these, please get used to them as soon as possible

Important Notes for All Labs About Attendance, Performing and Presenting the Work

1. You are obliged to read this document word by word and are responsible for the mistakes you make by not following the rules.
2. Not attending to the lab means 0 out of 100 for that lab. If you attend the lab but do not submit the preliminary part you will lose only the points for the preliminary part.
3. Try to complete the lab part at home before coming to the lab. Make sure that you show your work to your TAs and answer their questions to show that you know what you are doing before uploading your lab work and follow the instructions of your TAs.
4. In all labs if you are not told you may assume that inputs are correct.
5. In all labs when needed you have to provide a simple user interface for inputs and outputs.
6. Presentation of your work
   You have to provide a neat presentation prepared in TXT form for programs, and PDF form for reports. Your programs must be easy to understand and well structured.
   Provide following six lines at the top of your submission for preliminary and lab work (make sure that you include the course no. CS224, important for ABET documentation).

   CS224
   Lab No.
   Section No.
   Your Full Name
   Bilkent ID
   Date

   Please also make sure that your work is identifiable: In terms of which program (and other things) corresponds to which part of the lab.
7. **If we suspect that there is cheating, we will send the work with the names of the students to the university disciplinary committee. You can experiment with ChatGPT for learning; however, you cannot use/modify the code generated by it. Such an act is classified as plagiarism. Note that MOSS is capable of detecting ChatGPT code. Furthermore, remember that the code you use from ChatGPT can also be used by another student in the course. Make sure that the code you submit is really yours and has been internalized.**

DUE DATE PRELIMINARY WORK: SAME FOR ALL SECTIONS

NO late submission will be accepted. Please do not try to break this rule and any other rule we set.

a. Please upload your programs of preliminary work to Moodle by **13:30 on Wednesday, 27 November, 2024.**

b. Please note that the submission closes sharp at 13:30 and no late submissions will be accepted. You can make resubmissions so do not wait for the last moment. Submit your work earlier and change your submitted work if necessary. Note that only the last submission will be graded.

c. Please familiarize yourself with the Moodle course interface, find the submission entry early, and avoid sending an email like "I cannot see the submission interface." (As of now it is not yet opened.)

d. Do not send your work by email attachment they will not be processed. They have to be in the Moodle system to be processed.

e. For preliminary work, upload a file that includes all programs in proper order with the filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.pdf** Only a PDF file is accepted. Any other form of submission (Word file etc.) receives 0 (zero).

DUE DATE PART LAB WORK: (different for each section) YOUR LAB DAY

a. You have to demonstrate your lab work to your TA for grading. Do this by **12:00** in the morning lab and by **17:00** in the afternoon lab. Your TAs may give further instructions on this. If you wait idly and show your work last minute, your work may not be graded.

b. At the conclusion of the demo for getting your grade, you will **upload your Lab Work** to the Moodle Assignment, for similarity testing by MOSS. See below for the details of lab work submission.

c. Try to finish all of your lab work before coming to the lab, but make sure that you upload your work after making sure that it is analyzed by your TA and/or you are given the permission by your TA to upload.

Part 1. Preliminary Work / Preliminary Design Report (50 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 early branch prediction in this pipeline. Hence, the branch resolution is done in the Instruction Decode stage.). You are also expected to implement j and addi instructions. 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) The first page must start with Course No. etc. as indicated above. Prepare a cover page for your preliminary report PDF containing the following five lines (make sure you include the course no. CS224, important for ABET documentation).

CS224
Lab No.
Section No.
Your Full Name
Bilkent ID
Date

b) **[10 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) **[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 b above) so that your pipelined processor computes correctly.

d) **[15 points]** Write small 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. Write:

- A test program with no hazards (to verify that there are no problems with the connections in your pipeline etc.)
- A test program that has one type of hazard for each type of hazard you have identified.

In the end, have at least 4 test programs (testing at least 3 hazards) with their machine code (in hex)[1].

[1] We have provided you with an online assembler to easily assemble your MIPS assembly code to hex to put inside the instruction memory: https://bsahin.xyz/misc/mips/ If the website is unavailable, contact berkan.sahin@bilkent.edu.tr

You may check the samples given in "Sample tests for hazards.txt" for intuition. However, you must write your own tests for Part 1.d

e) **[20 points]** You are given a skeleton System Verilog code for your pipelined MIPS processor in the file ***PipelinedMIPSProcessorToFill.txt***. The places in the code that needs to be modified are shown with comment blocks above them. Note that the datapath, as well as the pipeline registers are mostly empty. Based on the pipelined datapath diagram we provided you (**PipelineDatapath.png**), write the SystemVerilog modules for the hazard unit **[5 points]**, PipeDtoE **[10 points],** and PipeEtoM **[5 points]**. Note that the names of the inputs/outputs on your SystemVerilog code must match the names used in **PipelineDatapath.png** where possible.

Part 2: Simulation and Implementation (50 points)

Implementation in SystemVerilog & Simulation (35 points)

You are given a skeleton SystemVerilog code for your pipelined MIPS processor in the file ***PipelinedMIPSProcessorToFill.txt.*** The skeleton code contains all the modules used in a single-cycle implementation, with some empty modules corresponding to the pipeline registers and the hazard unit. Note that the datapath is mostly empty as well. Two of the pipeline register modules are filled to provide an example implementation. Fill the pipeline registers and connect them, along with the register file, ALU etc. to the skeleton datapath we have provided. You don't need to follow the skeleton code point by point. You can modify the structure of the skeleton code as you wish, or implement the processor from scratch as long as it works. Note that in the skeleton code register file is initialized to 0 on reset. That effectively means all register values are set to 0 at the beginning.

Simulate and test your pipelined processor using the code you have written in part 1-D or the examples we have provided. Verify that all the instructions work correctly, even in the presence of hazards. When you have studied the simulation results and can explain, for any instruction, the values of PC, instruction, writedata, dataaddr, regwrite and the memwrite signal, then call the TA, show your simulation demo and answer questions for grading. The purpose of the questions is to determine your knowledge level and test your ability to explain your demo, the SystemVerilog code and the reasons behind it, and to see if you can explain 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.**

Testing on BASYS3 (15 points)

You should 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. Therefore, 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.] **You should use the values in the Execute stage of the pipeline for the ALU output and WriteData, and the values in the Decode stage for the control signals.** So, a new top-level SystemVerilog module is needed, containing top.sv 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 SystemVerilog model to the inputs (100 MHz clock and pushbutton switches) and outputs (AN, SEG and DP signals to the 7-segment display, and memwrite and regwrite going to LEDs) of the BASYS3 board and its FPGA. **Hint:** You can find a base constraint file for the BASYS3 board here: https://raw.githubusercontent.com/Digilent/digilent-xdc/master/Basys-3-Master.xdc You can modify this file to create your constraint file instead of writing it from scratch [You can download the file by right clicking and selecting "Save Page As…" or the equivalent option in your browser].

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 45-point demo and Oral Quiz, to determine how much of the 45 points for this part of Part 2 (implementation) is deserved, based on your demo, your knowledge and ability to explain it and the SystemVerilog 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 Lab Work for MOSS Similarity Testing

1. Submit all the SystemVerilog code used in your project (including testbenches and parts of the skeleton code we provided, if used) for similarity testing to Moodle.
2. You will upload one file. Use filename **StudentID_FirstName_LastName_SecNo_LAB_LabNo.txt**
3. Only a NOTEPAD FILE (TXT file) is accepted. No txt file upload means you get 0 from the lab. Please note that we have several students and efficiency is important.
4. *Even if you didn't finish, or didn't get the SystemVerilog code working, you must submit your code to the Moodle Assignment for similarity checking.*
5. 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 the code you submit is code that you actually wrote yourself! You are not allowed to use web resources that solves the assigned programs.

6. The effectiveness of MOSS for chatbot code is quite good, with a detection rate of much higher than 50%. (When the answer is provided by a chatbot.)

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. The questions asked by the TA will have an effect on your 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 works for ChatGPT code too. 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.