EEL 3701C - Digital Logic & Computer Systems

Lab 5 Report

Charles Richardson
73112398

Prelab Report

Prelab Questions

1. Left shift is equivalent to MULTIPLYING by powers of 2
2. Right shift is equivalent to DIVIDING by powers of 2
3. Logical shift right will replace the leading bit with a zero, which can jeopardize the value of the number if 2s compliment is used. Arithmatic shift right preserves the sign when shifted. Therefore, arithmatic shift right must be used when shifting a negative number
4. 256 possible operations

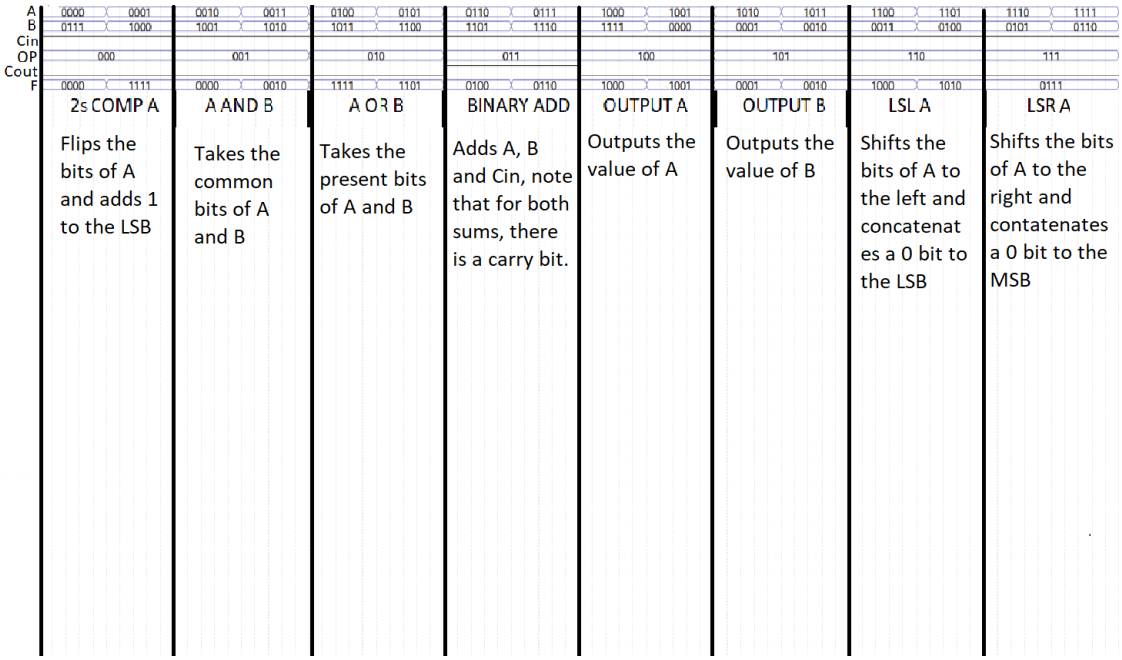
Prelab Design and Implementation

The design of the prelab was given, it was up to me to implement it. I knew that I needed to make a ALU, which consisted of a 8:1 mux. I reused implementations from previous labs to get this done, and adjusted the values to made it accurate. To make the datapath, I had to get curious and do some research, consulting various TA's to get the work done correctly.

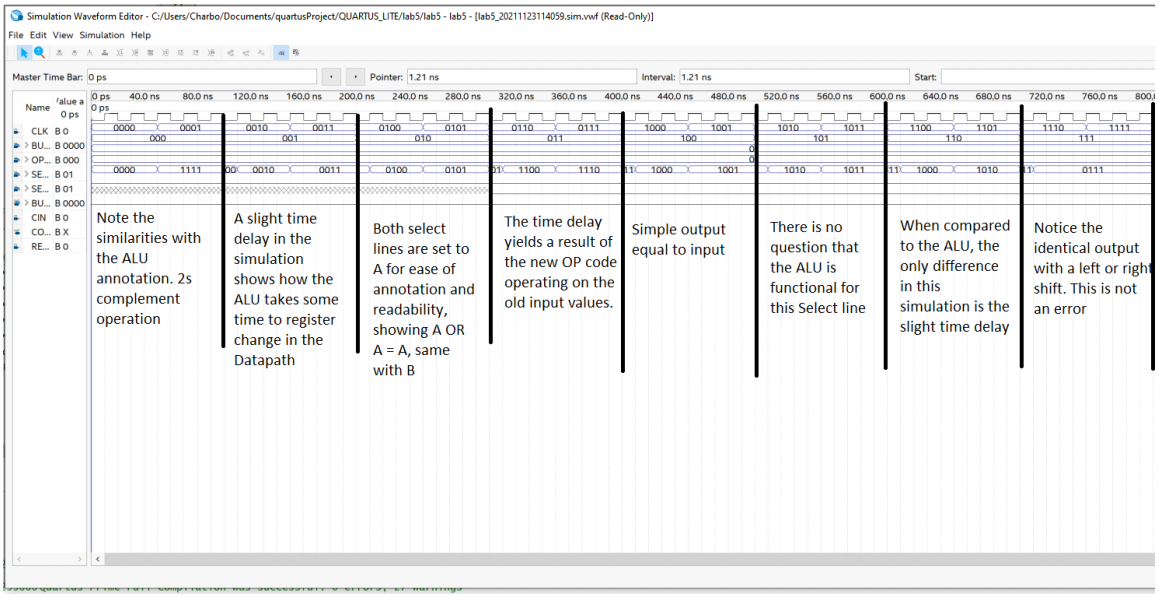
Reflection

Ultimately, the Datapath made working with the ALU slightly more difficult due to propogation delays and different select lines. Despite these challenges, I was able to successfully design the ALU and the Datapath with the help of many TA's and countless hours in front of the computer. I was proud to see the properly functioning system, but looking back, I am disappointed it took me so long to make.

Prelab Homework



Problem 1: ALU simulation with annotations



Problem 2: Datapath Simulation with Annotations

ALU and Datapath VHDL implementations attached as HDL files.

When submitting the Prelab Report, you do not need to include anything below this line.

When submitting the Postlab Report, please preserve everything you did in the Prelab Report (ie., append the Postlab Report to the Prelab Report), then submit it to the same assignment in canvas. The two combined is called the Lab Report.

Postlab Report

Problem Statement:

The purpose of this lab was to provide students with experience designing a simple processor. Once fully assembled, the processor will accept a 9-bit code and a start signal and return a result which has been operated on in a specified way. The system can perform a number of computations with very little overhead from a user perspective. They simply need to know the syntax of the 9-bit code and the possible operations.

Design:

The majority of the design decisions were made for me with the provided templates. I was indicated to build an ALU, then make a datapath around the ALU. The ALU is within the datapath, which meant I had to design it such that it could be used without directly inputting to it. This requied some signals to avoid any issues with inputs and outputs overlapping.

Next, I was told to build a controller to simplify the input and output of the system. The controller was a seperate entity from the datapath which output values that the datapath took as input, and vice versa. This required signals for the same reason as the Datapath and ALU relationship. I chose to use port maps to make the connections in the processor simply because I do not know an alternative way.

Implementation:

Implementation was guided with templates, but it was far from simple. Building up from the simple components, the project quickly became convoluted and required some cleanup. Using the ASM drawing as a guide, I implemented each case as specified and then built out the connection of the datapath and the controller using portmaps and signals to keep things tidy. Overall, the implementation of the complete system took me well over 1 week, given I started working on it before Thanksgiving. In hours, I predict that the project took me 30 hours.

Testing:

To test the processor, I began from the simplest components and worked my way out. I first made sure that the ALU worked. Once I was sure it was functional, I set the Datapath to my top level entity and simulated that. The Datapath revealed some strange propogation delays, but a TA told me that this is expected, yet it should be mentioned. The delay was causing a newly triggered OP to perform operations on the previous register values, despite being triggered to a new value at the same time as the OP code. This would only last one clock cycle, so it seemed to be a minor issue.

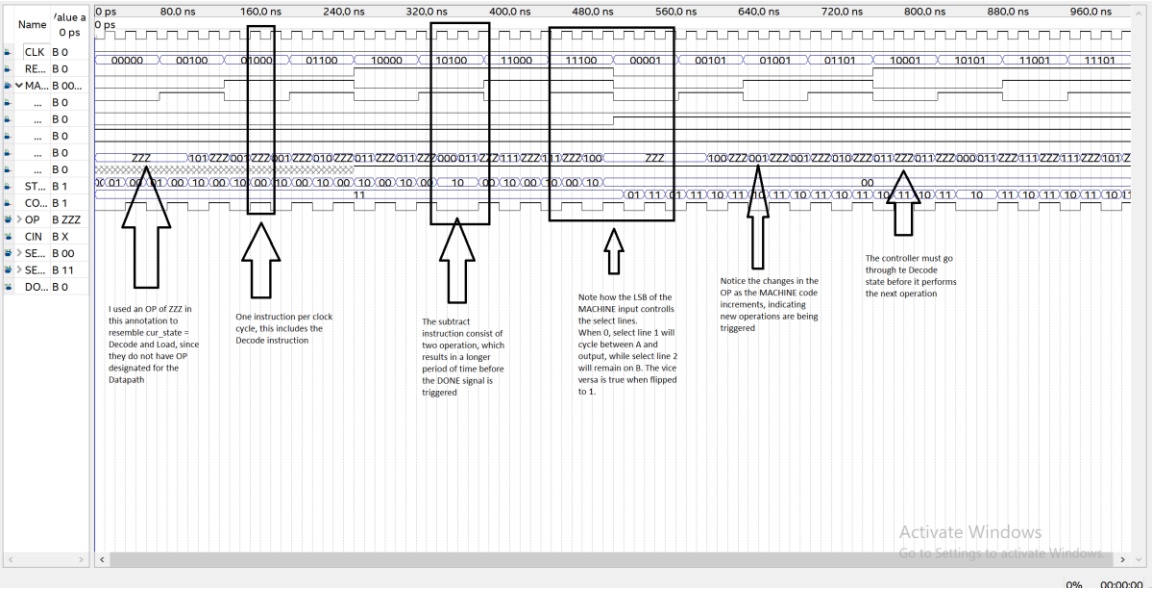
For my controller, I ensured that the proper OP and select lines were being triggered under various inputs, and it all checked out. Finally, moving to the top level implementation of the entire processor, I first tried implementing every possible input, but realized that 2^8 inputs would be hard to read. When I tried one input at a time, they returned the right input, but it was

after 3 or 4 clock cycles. This confused me, and after hours of debugging, I was unable to solve the problem. In retrospect, I believe that part of the problem is coming from the propogation delay in the Datapath, but I do not have time or resources to figure it out. Screenshots of the issues are included in the **Prelab Homework** section and the **Appendix**.

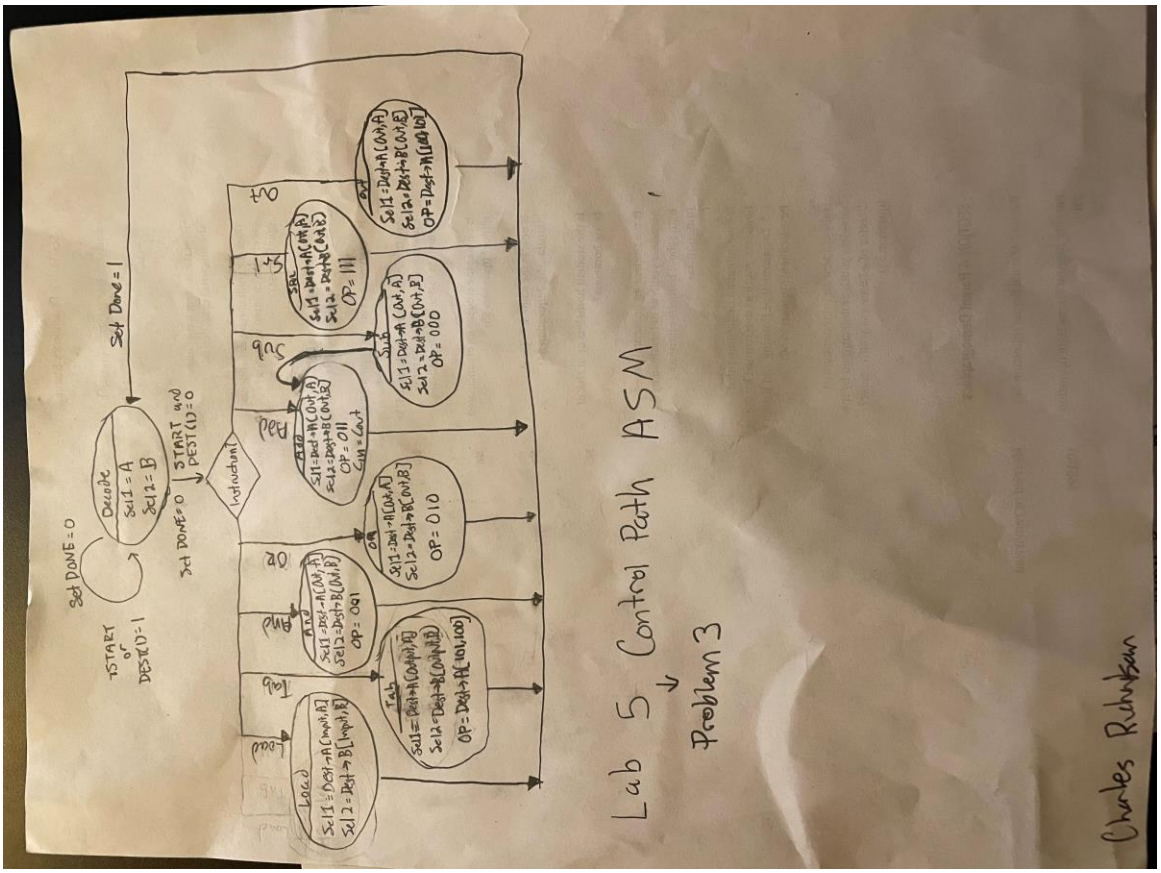
Conclusions:

In conclusion, I implemented a semi functional processor. I realized slow success for the first parts of the lab, and quickly got dragged down mentally as the lab progressed. I spent 3-4 hours on a problem to eventually be told by a TA that it was not required to be solved. This lab took a lot out of me, and I am leaving it slightly unfinished due to other requirements. In the future, I plan to spend more time at in-person events to develop better connections with my peers, TAs and professors so that asking for help demands less friction.

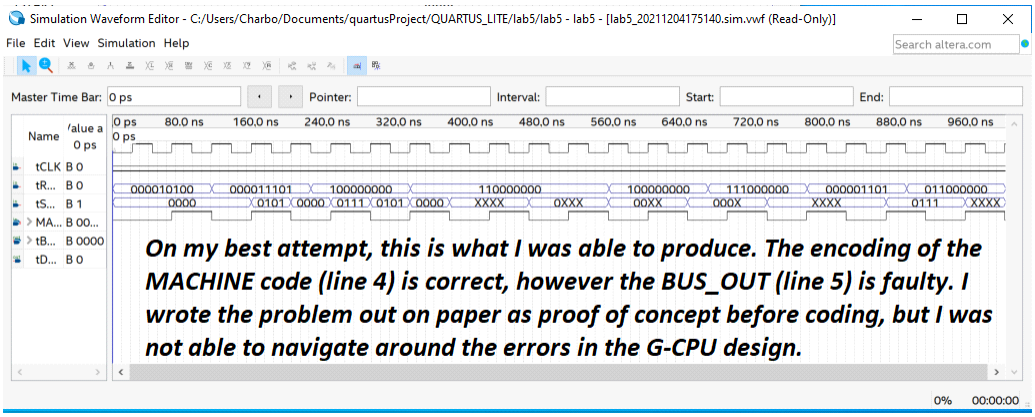
Appendix



Controller simulations with annotations



Controller Path ASM (unable to rotate for some reason)



Processor simulation with a comment on the state of the system

Problem 4 Part 2 Logic written out with Part 3 included below (unable to rotate for some reason)

Controller and Full Processor VHDL implementations attached as HDL files.