Joshua Andrews

ECE 473

12/13/17

Project Report

**5-Stage Pipeline Processor**

Introduction

Through the course of the semester we first developed modules for and then implemented a 5-stage pipeline processor. After over 150 hours, countless late nights, and a still unsettled hatred for the Quatrus bugs and compile times, a successful five stage pipeline processor was designed. Two successful test programs were required to receive full credit and five were passed. An updated, easier to edit version of the charengine.v file for the VGA module was also created to count for bonus points and to hopefully give students next year an easier time debugging.

Design Skills

The design and implementation of the pipeline processor was extremely challenging. The design was even more challenged by the fact that I was in a single person group. While I did get some relief in the amount of test programs I need to pass in milestone 4, the extra effort of having to design the entire project by myself severely outweighed that fact.

The initial problems encountered were purely software based. Initially it took some effort to learn how to use the Altera software correctly. It also took some time to learn how to write proper Verilog code. Once those two hurdles were tackled it became much easier to complete the labs.

Another skill I learned through the project was to keep the naming conventions of labels consistent. Once I started the milestones the number of labels grew substantially and trying to keep track of them all, especially if the labels in the same stage of the pipeline were formatted differently. Using consistent labels made changes to the design easier to make and to keep track of.

While most of the other students in the class switched to just using labels with no wire connections between blocks, I used them as much as possible. It allowed me to highlight a wire and quickly see every connection it made to every block in my design.

Finally, I learned to thoroughly debug the test programs before making changes to save time compiling and hopefully prevent changing something that does work. Going cycle by cycle through the test program and comparing it to the MARS MIPS simulator results to see exactly what was happening made it easier to isolate the bug and correct it. I also think I spent much less time waiting for compilation to finish, especially in the final two milestones, by using this method.

Project Status

The processor as designed was successful in performing five of the nine test programs provided. When I reached milestone 4 several of the test programs were almost working simply based on the completed milestone 3. After many hours of debugging I was able to identify and correct an issue with the jr instruction. After I made that fix is when I was able to get five test programs to work. The other four tests involved the stack pointer and I was unable to fix that problem with the processor before the deadline.

Debugging

Ten percent of the time was spent writing code, 90% was spent debugging. I found the code for all the modules not hard to formulate but ensuring all the proper signals were set and sent to the right place was the difficult part. VGA, MARS, and the onboard leds were used in the debugging process as it allowed me to identify how the instruction was being handled by the processor and to verify in MARS that it was correct.

The most helpful tools by far when it came to debugging was the VGA interface and MARS. Using the two together allowed me to pinpoint exactly what was going wrong and where. Some of the signals that I pretty much always kept track of was the current PC and instruction, ALU operands, and the write data and destination. The LEDs were used for 1-bit control signals and I had one for jump, jr, branch, write enable, and data memory write. I could practically see how the instruction was being piped through the processor. I modified the VGA file and will talk about what was done to it in the next section.

Purchasing my own board was extremely helpful to the design and debug process. Having my own board allowed me to work on the project at any time. I also did not have to waste valuable time in lab compiling various changes to the project as I could make and test changes at home. Coming into lab with a design that was almost working made for a more productive use of that time as well as to better help the TAs give advice to what that problem might be. It also gave me the added benefit of becoming more familiar with the board which I believed helped me in the debug process.

VGA

The VGA interface developed by Tristan Woodrich was by far the most helpful debugging tool. For a bonus and as a debugging aid I worked a lot on setting up the VGA module to make it easier to see how the processor was handling the instructions and to easily keep track of all the data with the use of labels on the monitor. As to keep track of which milestone and test number each sof file corresponded too I added a spot for the lab members names, and which milestone/test it was. This also helps instructors/TAs to see what group and what test is being performed very easily.

To decrease the complexity of and the time involved in changing a label for the GP Registers column, I had to implement variable length bit arrays that would store the string (label) to be displayed. A simple for loop and shift mask then assigns each character (8 bits) to the proper position in the hex buffer for printing to the screen. Unfortunately to get the variable length arrays to work I had to use a separate parameter value for each string length and then multiply that by 8 bits. This requires and update to both the charengine.v file (to change the label) and then the VGA block diagram parameter list with the character length of the new label.

The labels are entered by the user in string form and then maps the ASCII values of each character to an array. The original charengine did not follow the ASCII convention and I had to include ASCII support for at least uppercase characters for the array technique to work. This was due to one of the problems I encountered with Quartus, which didn’t formulate the display correctly if I included more than one if/else/loop under a case statement. I had to keep the original mapping of 0-F, along with the ASCII mapping, for the data and labels to display correctly. I include support for uppercase characters, numbers, and a couple special characters in the labels.

Because the GP register column was instrumental in my debugging process I decided to double the amount of them avail on the display to allow keeping track of even more points in the processor. All of the additional registers have easily editable labels as well.

Comments in the new charengine file and a new section in the VGA readme should make adjusting the labels self-explanatory and provides them with a couple tips to tweak the display if things are not displaying properly.

Suggestions

The overall project as well as the labs through the semester are well laid out. The labs are a very good and well-paced way to get familiar with the Altera Quartus software and Verilog. Getting familiar with how to make modules through the labs was also very beneficial when it came to the design of the 5-stage pipeline processor. Having some modules and a very basic layout of the processor going into the final project was a huge help. While the workload for the final project was substantial, having worked on the “template” beforehand helped, a little, to reduce the amount of work.

One of the biggest things I noticed, when I got to the final milestone, was that some of the instructions I had previously tested in earlier milestones, no longer worked. To combat this in the future I would suggest making the test programs for those milestones a little more comprehensive. This will obviously require additional effort to complete the first two milestones but the workload for them was small when compared to the final two. For the final two milestones I had to go back and fix instructions that no longer worked. I believe increasing the robustness of the test programs for the first two milestones would help to balance out the workload.

There was also discussion in class about splitting milestone 3 up somehow to help make the workload more balanced. I believe implementing the new test programs while keeping the deadline of milestone 3 at two or three weeks would balance out the workload without having to change the structure of the project.

While my updated charengine.v for the VGA module is aimed at making design analysis and debugging easier, I think I learned the most about the processor when things didn’t work.

Conclusion

While the amount of work involved in designing, and then getting to actually work, the processor was substantial, it was also very interesting and informative. I really enjoyed being able to make a processor and all the things that went into it. The TA’s and Prof. Yifeng, as well as Nick Lajoie and Dan Schalbig, gave me advice along the way and that really helped when I was just absolutely stuck. This class really made me realize that I did in fact make the right decision to double major but also to keep electrical engineering as my primary. All joking aside this was a great class and a great project and I found that the class helped me with the project and the project helped me with the class.