Brandon Koonce

ELEC 5200

Project Report

The main things I learned from working on the project is that a doctorates is not required to build a simple working processor and also a better understanding of why assembly languages are written the way they are. At the beginning of the semester, when the project was first assigned, I had no idea how I was going to actually build a working processor, but the way it was broken down into steps made it manageable and almost simplistic when looking back. As for building the parts of the processor, the only component that I had not built in Digital Design was a control unit. The control unit was not hard to construct since we were required to create a diagram with all the signals that corresponded to the instructions, made it manageable to translate into VHDL. The other main thing I learned was how the assembly language actually related to hardware. In Computer Systems we learned how to program in assembly but I did not fully understand the reason why certain commands were set up the way they are. After designing the hardware and the instruction set it is clear as to why most commands only work with registers to counter Von Neumann bottleneck or how branches actually work and jump to the labels.

If I could redo the project I would change my architecture to a multi cycle. Single cycle seemed easier to design, but when trying to place the design on the FPGA I ran into problems with clock skew. The reason I believe multi cycle would not have produced as many problems is due to having multiple stage to clock data compared to one cycle meaning registers have to be read and written to in addition to reading the actual instruction causing a plethora of clocking issues. Even if I used single cycle again I would have designed my second ALU to properly update the program counter with the address rather than fixing the issue by adding more hardware.

Some advice I would give to future students is to keep the data path and instruction set up to date. It is easy to work on the VHDL and plan on going back updating them but never getting around to it. The other main thing is even if everything simulates perfectly in ModelSim do not think that downloading to the actual FPGA will run flawlessly. Some of the issues that can cause this are when things simulate but are not actually loaded as expected but other issues can be caused by timing problems. Even when using single cycle with a exceptionally slow clock speed timing issues can still arise due to excessive latches.

My single cycle processor did not function on the FPGA due to timing issues. The main cause of this is due to reading the instruction RAM on the rising edge, accessing the data RAM on the rising edge, and storing the register values on the falling edge. The clocking design of the RAM is flawed, but I was unable to find a way to change since the generated RAM file included other files I was unable to locate. As for the register file, I made it save the modified register on the falling edge believing the data could be processed when the clock is high assuming a long clock. With only three components being clocked as such, the processor would at least generate some data even if invalid. The fact that no data was produced led me to connect the program counter to the LEDs and observe that the program counter never updated. After witnessing this, it is possible the design synthesizes as intended in ModelSim, but is incorrectly constructed on the board, however no variables or questionable statements were used. Farther evidence that discredits this explanation is originally before the two RAM files were implemented, when the instructions were manually entered through switches, the processor produced values, unfortunately the code was modified to fix incorrect instructions and to troubleshoot the later timing issues which made it impossible to revert the changes. In short, the processor failed to function on the FPGA, I believe, due to timing issues even though the clock was connected to a push button giving the data path ample time to be traversed.