Assignment 4 Report

Nicholas Petosa (Partner William Ma)

Design Decisions

In order to increase execution time for fmedian2, we implemented a data forwarding controller in our processor to replace the stall controller we had previously. We saw an improvement in execution time from 1min 6sec with stalling to 52 seconds with data forwarding. We were still above the 45 second execution goal, so our next strategy was to increase clock speed by modifying PLL. We iteratively increased clock speed and recompiled the Verliog until we started to receive negative slack messages in TimeQuest. We found 90 Mhz to be the clock speed with the most stable positive slack, so we went with that. The result was that fmedian2 executed in about 26 seconds, a huge improvement from the 52 seconds experienced prior. This large increase can be attributed to nearly doubling our clock speed, which halved execution time.

We then implemented I/O devices. For HEX and LEDR, adding load functionality was simple. I made SW store to a data register as well as the displays. Then, I simply added a ternary statement in the data bus to check if a MMIO LW request was being made to either HEX or LEDR, and then simply returned the value from that data register. For the SW and KEY, we needed more complicated devices since they both needed a data register and a control register. This device was implemented separately as a module called Debouncer, as it supported control registers as well as optional debouncing, which is necessary for the SW. For KEY, the debounce value is set to 1 to skip debouncing while allowing us to reuse all the LW and SW MMIO capabilities afforded by the device. Debouncing is implemented with a simple counter that increments every pulse, resolving in a total of 10ms. For the timer, another module device was added. It is a simple counter but also has support for getting and setting the limit, counter and control registers with MMIO calls.

Finally, we wrote the assembly for xmax.asm, which uses the timer functionality to pause between LEDR states. The KEY reads are implemented in such a way that they can change the period speed asynchronously to the LEDR display. We do this by skipping the KEY read logic if the button has yet to be released. The keys either increase or decrease TLIM by a factor of 250 milliseconds, with a matching numerical display on HEX. Additionally, we only consider the first two bits of KEY to so that we exactly achieve the functionality put forth by assignment 1.

Problems/Issues

There were some challenges in implementing data forwarding, as we had trouble converting from our stalling strategy to a forwarding strategy. Ultimately, reworking our dependency trigger logic gave us the desired results. Another challenge was deciding what parameters we needed for our devices, as there was a lot of functionality that needed to be achieved in just one module (the ability to discriminate between reads and writes, as well as what we were reading and writing from, as well as whether that operation is allowed in the first place). This took a lot of trial and error on my end. In the end, I tested the device functionality for debouncing SW and reading KEY and was successful in all of the MMIO operations. The same can be said for timer.

Contribution: 50%

I implemented the I/O devices and timer as specified in the lecture slides, and I wrote half of the assembly for xmax.asm.