Project Status Report #9
David Gronlund
Group A3
4/20/19

This week I worked on trying to connect the core in the Vivado block diagram, add a memory interconnect to the core using AXI, improving the timing of the core, offer a standard-io feed from the core, and then work out more details of how we are going to support vectorized floating point.

The first problem I solved for the week was writing a converter between the simple memory interface the core currently uses for instruction and data memory, and the AXI protocol used by the Xilinx IP catalog. The converter buffers all of its output streams to avoid having a combinational loop, which made use of the new flow controller logic I wrote last week.

I had to take the core and appropriately wrap it in normal verilog and then package it through the Xilinx flow, but after a lot of syntax tweaking I was able to add it to a normal IP design and connect it to the supervisor ARM core in the IP packager. This is flow we will be using to test the core on the FPGA, so getting this working was an important step to our FPGA testing as well as further timing benchmarks in a real memory interconnect.

To improve the cores timing I broke the register status file into two separate registers which now only require a single write port each, which means distributed RAM can be used in the FPGA and the footprint/timing becomes better. I also put register buffers at the the end of both the decode and execute stages instead of normal registers, which while slightly increasing the register usage of the core, improves the critical path of the ready signal which was starting to become a problem with increasing usage of interconnect to access the core's memory from the outside.

I broke out the standard output I had been using in simulation only for testing prints, and instead made it a full fledged stream which could be hooked up to an AXI-FIFO, which allows the core to be tested with printing on the actual FPGA.

We sat down and narrowed down exactly what vector instructions we are going to support. We knew already that we were only going to support 32-bit floating point, but we also got rid of a lot of the vector load/store instructions, since our access model is going to be very simple due to the limitations of the FPGA internal memory.