Project Status Report #6
David Gronlund
Group A3
3/30/19

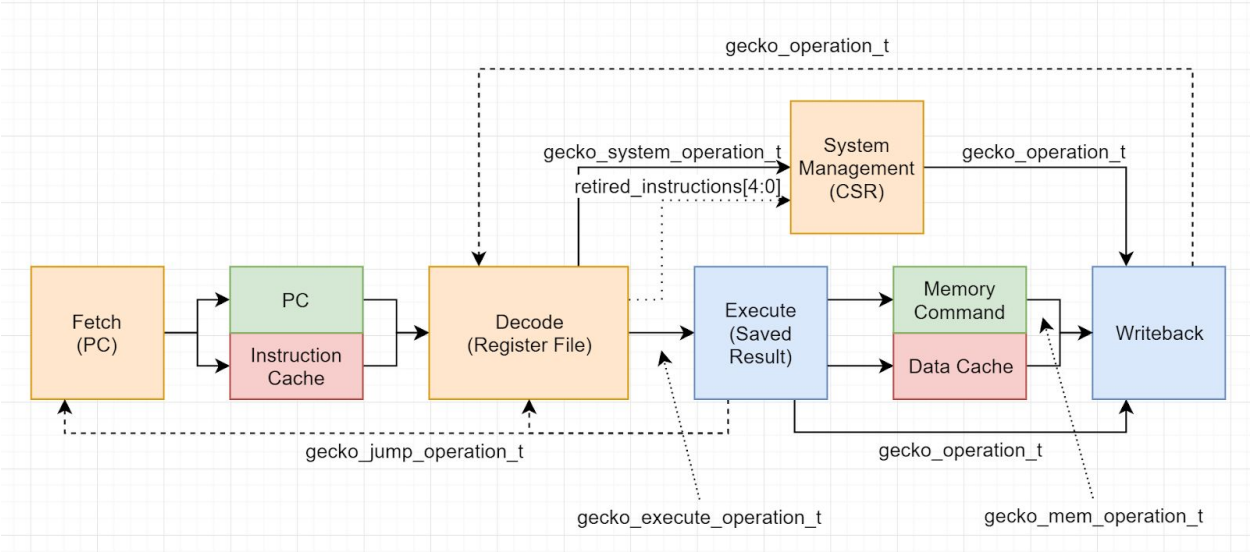
This week I worked on improving the CPU performance and tweaking its design so that adding a floating point unit would be easier for us, as well as working on bringing up the IPC, and bringing up the frequency of the core. Some things of note for the IPC were that our benchmark, Dhrystone, is always compiled with -O3, which I had forgotten to include, and just doing that improved our benchmarked performance from about 0.4 to 0.5. Separately I made a few different changes to get the CPU to go faster, which got us from about 220 MHz to about 280 MHz, on our low-end FPGA.

The first thing I did to improve frequency, at the cost of IPC, was to add an easy way to pipeline the instruction and data memory stages with arbitrary amounts of stages, as well as infer using the output register of the block-rams when two or more cycles of latency were used. This was the only major change that was needed to improve the frequency, especially given the complexity of the decode stage and the memory architecture of the block-ram. Our critical path is still in the decode stage, but now it is in the register status logic.

Next to improve the IPC I changed the logic to support arbitrary out-of-order retirement, rather than previously just bottlenecking write-after-write instructions through the execute stage and just refusing to issue them through other stages. This change even simplified the decode stage logic, which improved the critical path ever so slightly, at the cost of having to keep an extra status register in the write-back stage.

With all register writeback commands having a tag associated with them from the out-of-order retirement modification, arbitrary forwarding was now easy to add, as I gave the decode stage an arbitrarily sized array of forwarded results that could be given to it from any bus in any order, that could now be used for forwarding results from anywhere else in the pipeline. Since the FPGA fabric may provide timing issues with forwarding too many results, I specifically made this modification easy to modify or undo if forwarding hurts frequency more than it improves IPC.

The last modification I did this week was to add branch prediction, which was the last big IPC improvement we wanted, and I gave it enough parameters to store a different sized branch-target-buffer for flexible configuration later, trading area for IPC. The predictor just uses a two-bit saturating counter right now, but the prediction functions are compartmentalized so that the user could easily write a shift-register history predictor or anything else in the future. This feature works, but there is a bottleneck in speculative execution I need to fix before this improvement could really help.



Updated pipeline with Branch Prediction