Logic Type Rock
MP4 CP1 Progress Report and Road Map
dbycul2, danielp7, jayhp2

Progress Report

Basic RV32I Pipelined Design:

- Cache/Arbiter:
    - Design (dbycul2): For CP2, we used the provided cache and the cacheline adaptor from mp1 in tandem with our new arbiter to provide a basic caching system for our CPU. The arbiter prioritizes instruction cache accesses and uses a three state system to ensure that both caches aren't requested at the same time.
    - Verification (dbycul2): We used our datapath from CP1 and the test code for CP1 to test the cache design. A simple stall signal was introduced into the datapath to wait for the no longer instant memory response. We then verified that all of the register values matched the expected values.
- Forwarding:
    - Design (danielp7): Daniel used the forwarding design done from the previous checkpoint and implemented it into the project. This was done by creating a new forwarding module and connecting the components in the datapath and into the control unit. The main focus of forwarding is to forward register values early into instructions that may need it instead of having it wait for the final register value to be filled. This allows for faster execution.
    - Verification (all): Everyone worked together to debug the waveform based on the final register values. This was able to be done because the cache/arbiter was implemented and debugged before the rest was added for debugging purposes.
- Hazard Detection:
    - Design (jayhp2): Created a new hazard detection module with logic taken from the textbook. Checked to see if the instruction is a load. If it is, then we check to see if the destination register in the EX stage matches either source register from the ID stage. If this condition holds, then we stall by adding nops in our datapath.
    - Verification (All): Everyone worked together to debug our hazard detection with forwarding with the testcodes given. We looked at the waveforms of competition code and the checkpoint tests to verify the design.
- Branch Predictor:
    - Design (All): Everyone worked on making a static branch not taken predictor that flushes the pipeline upon incorrectly branching. The design was taken with queues from the textbook and compares the
- Advanced Features:
    - Design (jayhp2): We decided to go with three advanced features and potentially four if we have time. The first is basic hardware prefetching, second is memory leapfrogging, and third is pipelined L1 caches. I created the designs based on our own datapath and how it could integrate with our current setup.
- Hook up Shadow Memory and RVFI Monitor:
    - Everything (Dawid): Dawid was responsible for connecting and activating the RVFI monitor and Shadow Memory. This included making monolithic registers to capture the correct values for the RVFI monitor along with setting those values at the correct point in the pipeline.

Road Map

1. Advanced design (Implementation and Verification = all)
    a. Pipelined L1 Caches (6)
    b. Basic Hardware Prefetching (4)
    c. Memory Stage Leapfrogging (12)
2. Extra design (Implementation and Verification =all)