CP3 Progress Report

Work breakdown and implementation of functionalities

This checkpoint involved us programming the advanced features for the pipeline. Peter coded the parameterized cache and pipelined cache. The parameterized cache allows the user to adjust the number of sets and number of ways in a cache. Potentially the cacheline size can be changed as well, but it requires changes in source_tb. The pipelined cache allows the cache to receive new requests while it is responding for the previous request, so that the cache doesn't need to stall on consecutive hits.

Lakshya coded the branch history table and the branch target buffer. The BHT and BTB are both parameterized for the number of sets. Currently, the BHT uses 2-bit saturating counters for the local branch prediction. It defaults to Not Taken since it doesn't store any tag/metadata to check if an instruction is a branch/jump. This allows non-branch/-jump instructions to continue executing without mispredictions. The BTB stores the predicted target address whenever a misprediction occurs.

Mridul coded the M-extension and the eviction write buffer. The M-extension is a logic based multiplier and divider (with both a signed and unsigned version); an eviction write buffer is used to hold dirty evicted blocks in a cache, so that a CPU can receive the evicted data faster rather than waiting for a dirty block to be overwritten to first.

Testing Strategy

To test our design with the advanced features, we used various assembly test code, some that was provided, and some that we created ourselves. The main assembly that was helpful for us to use to test was the mp4-cp<x>.s and comp<x>.s. These few test cases were important because they were very long and much more exhaustive.

For our own testing, we used our factorial.s test code from previous MPs to give ourselves a shorter code that was easier to follow through and debug in the waveforms.

Additionally, we made sure to attach the RVFI monitor and Shadow Memory modules as well as we could since those are very helpful in automatically checking our design functionality as it executes.

Roadmap

Our plan moving forward for the next week, which is the design competition, is to test various configurations of the caches and predictors to see what gives us the best performance and power. We will use the provided competition code to profile our design for performance and power using the Timing and Power Analyzers of Quartus.