ECE411 CP1 Progress Report

Progress

For checkpoint 1, we spent a lot of time learning about the general implementation of the out-of-order processor and discussed our desired baseline implementation and functionality, keeping in mind some advanced features.

At this point, we have implemented and verified the fifo module that we will be using for various components in the processor (like the ROB, RRF, Free List, etc.). We also implemented and verified a cache line adapter for the instruction cache and data cache to interface with the new burst dram.

Additionally, we added a line buffer in our cache line adapter that currently only supports reads as we plan on implementing a pipelined cache by the time we need data cache (cp3). We also implemented a fetch stage for our processor that feeds the pc into our instruction cache and puts the cache read data (inside an instruction packet struct) into an instruction queue which is an instantiation of our fifo. This instruction queue will then feed instructions to our decode stage. The functionality of our fetch stage was verified by confirming correctness when using a small assembly program. We also planned out our entire processor design by making a block diagram.

Design Decisions

For the upcoming checkpoints, we have decided to go with explicit-register-renaming (ERR) architecture, as it seemed more intuitive for our group, and also allows for more flexibility and interesting advanced feature implementations.

Furthermore, we have added a line buffer to our cache, in order to get ideal zero-delay response if reading from the same line, there is a high chance we will get rid of this functionality since we are planning on having our cache pipelined fairly early on (ideally starting next cp).

Moving on, although not finalized, we have discussed the implementation details for various modules associated with ERR, we plan on using a modified FIFO, that keeps the index of the oldest, and the most recent entries, as well as having the functionality to be indexed into, for ROB, and RRF.

In the fifo, we decided to use the most significant bit (MSB) of head and tail for full/empty differentiation rather than the head and tail pointer positions to avoid edge cases.

Lastly, we have discussed future, more specific, design decisions, such as the sizes of RS for each functional unit, where we have decided on merging RS for MUL and DIV, and have the RS for MEM be the largest.

Work Split

Our work split was fairly even, where Ahmed worked on implementing and verifying the cache line adapter and Fetch, Eddie worked on implementing and verifying the FIFO and Fetch, and Mahir worked on implementing and verifying the Line Buffer and Fetch.