ECE 411: mp_ooo Progress Report

Checkpoint 1:
Akhil: Draw a design block diagram (using draw.io or similar, not handwritten!) of the basic OoO datapath [5]
Sam: Implement a parameterizable queue (depth and width) [5]
Ryan: Implement instruction fetch and integrate it with magic memory and your parameterizable queue [7.5] (Your program counter should be initialized to 0x60000000)

We tested the parameterizable queue and fetch through two test benches, fifo_tb and fetch_tb. In fifo_tb, we filled up the entire queue to the size of its depth to check it does not push to the queue when it is full. Similarly, we popped the entire queue to check it does not read anything when the queue is empty. We also test concurrent push and pop and if the queue becomes empty on reset signal.

In fetch_tb, we first check the implementation of instruction fetch and its integration with magic memory by checking the value that is being read/popped from the queue. We also use ordinary memory to test if the queue gets populated correctly when fetching instructions at varying cycles. The fmax of this design is approximately 1.534GHz as the minimum clock is 0.652ns.

Roadmap for Checkpoint 2:
Akhil: Integrate the provided multiplier into your processor as a functional unit [5]
Ryan / Sam: Your design must support out-of-order execution of the arithmetic operations (you must implement ROB, RS, CDB, etc.) [5]

For checkpoint 2, we will use the provided multiplier and make changes if necessary to integrate into the processor. We will test the implementation by writing test cases for multiply operation similar to what was done for mp_pipeline using RVFI. Out-of-order execution for arithmetic operation will be implemented through ROB, RS, CDB, RRF, and RAT. RVFI