Progress Report

Eric

- Created Control Word struct type – only used signals relevant to the control ROM in the control word.
- Made Control ROM – used the MP2/MP3 control file as a reference, and rather than creating a FSM with every signal, I only used the signals manageable within the control word.
- Created basic skeleton for Datapath and helped teammates match signals to their stage registers.

Gerardo

- Help create stage registers and wired them together in the datapath
- Updated design diagram as we went along to ensure cleanliness
- Ensured units in datapath and stage registers were wired correctly

Bryan

- Help create stage registers
- Help create datapath
- Update the datapath sketch and made sure the units were using the correct control word for each stage.
- Made muxes

Functionalities

Runs all basic RISC-V instructions (excluding data/control hazards) in a pipeline.

Testing Strategy

We started with just getting all the basic code down, to a point where it *should* be correct, and simply debugged on EWS using a combination of attempting to run the test code, and using verdi to check our signals/registers and whether they matched the spike values. Since we employed a divide and conquer strategy for programming the code, and debugged together as a group, it gave a lot of insight into the functionality implemented by each of our groupmates, rather than only knowing about the parts we implemented ourselves.

Timing/Energy Analysis

See .rpt files

Roadmap

Functionalities

- Integrate L1 cache
- Arbiter
- Hazard detection & forwarding
- Static branch predictor
- RVFI
- comp1 to comp3 & coremark (if time allows)

Who Does What

For this specific checkpoint (CP2), we plan to meet up and tackle each to-be-implemented functionality one at a time, together. We think having everyone in the group's opinions on the implementation methods is valuable for this portion of the MP. We plan to start by implementing the caches and making sure those work correctly before implementing the arbiter. Finally when we see it works the same as CP1, we will move on to handling hazards so that we can execute test code without NOPs.