EECS 470 Project Proposal Group 22

NOTE: References to team members removed for confidentiality and academic integrity.

1. Base Design

We have chosen to implement a MIPS R10K style architecture for our out-of-order CPU design

2. Advanced Features

1. Good unit tests (basic)
2. 2-way Superscalar (3-way if time allows) (difficult)
3. Good branch predictor (basic)
4. Early Branch Resolution (difficult)
5. Associative caches (basic)
6. Non-blocking cache (basic)
7. Instruction Prefetching (basic)

3. Project Milestones

For **Milestone 1**, we aim to implement the ROB, possibly with a parameterized number of entries depending on complexity. This will be the primary module we focus on. Additionally, we plan to complete the interfaces for the reservation stations, map table, and architectural map table. The ROB will be implemented collaboratively by the entire group, following the MIPS paper and lecture guidelines during our regular meeting times.

For **Milestone 2**, the main new features will be a working branch predictor and a 2-way superscalar implementation. By this stage, the modules for the reservation stations, map table, and architectural map table will be complete, allowing some programs to be executed. Control hazards are expected to be resolved by this milestone, and a skeleton for early branch resolution will also be developed.

For **Milestone 3**, the associative cache, non-blocking cache, and instruction prefetching will be fully implemented, and all hazards will be resolved. Early branch resolution will also be completed. Processor performance optimizations will begin at this stage and continue through the final deadline.

4. Project Organization

Most high-level processor integration will be completed collaboratively as a group. Individual modules will be assigned to team members, with frequent progress checkpoints to ensure that work stays on track and no one falls behind. Tentative module assignments are as follows:

- Branch Predictor: Not Listed
- Early Branch Resolution: Not Listed
- 2-way Superscalar: Not Listed
- Memory Components: Not Listed
- Instruction Prefetching: Not Listed

Responsibilities such as scheduling meetings, taking meeting notes, and checking in on progress will be distributed among the team to keep the project organized.

Module interfaces will be developed during full-group meetings. Once the interfaces are agreed upon, each module task will be delegated to a small subgroup to implement the required I/O, adhering to expected clock timing and bit stability.

---

5. Group Charter

Time Commitment: 15 hours/week per team member

Most work on individual modules will be done alone or in pairs. Weekly meetings will be used to sync progress and integrate modules.

Meeting Times:

Wednesday 3–6 PM at the Duderstadt Center
Sunday 4–4:30 PM virtual check-in

Unavailable Times:

The team will be unavailable during spring break. Otherwise, no major commitments are expected to interfere with project work at this time.

Unit Testing:

Each individual or subgroup is responsible for creating and thoroughly testing unit tests for their assigned modules before deadlines. This ensures that code reviews during general meetings are productive.

Communication:

- The team will use a Slack workspace with a general channel for announcements and scheduling.
- Each new task will have a dedicated channel for discussion among task members and to pin progress updates.
- Meetings, both pre-scheduled and emergency, will be communicated via Google Calendar and Slack; emergency meetings will also be announced via email.
- Team members are expected to communicate issues with deadlines, attendance, or task progress at least 48 hours in advance to allow adjustments.

Individual Strengths:

- Strong knowledge of Verilog semantics and low-level organization
- Experience integrating multiple systems and maintaining organized code
- Skilled at connecting modules and ensuring correct functionality
- Strong debugging skills and ability to break down code and systems
- Understanding of theoretical concepts and system-level integration
- Ability to dive deep into system integration and explore topics in depth

Individual Growth Goals:

- Improve the ability to evaluate multiple implementations and choose optimal solutions based on complexity and performance
- Gain more experience with SystemVerilog and implementing complex hardware in an HDL
- Enhance skills in connecting modules and verifying correct interaction
- Develop deeper proficiency in Verilog and understanding processor components

- Improve implementation of theoretical concepts and writing effective test suites
- Gain experience in independently developing solutions and understanding the project ecosystem (e.g., build tools, scripts, and workflow)

Signed: Team Members