EECS 470 Final Project - Fall 2025

The final project is to build on the VeriSimpleV RISC-V pipeline from project 3 to create an out-of-order processor based on the designs we've gone over in class. Projects will be done in **groups of five students**. All groups are required to implement an out-of-order processor with the base features listed below. Each group will also implement multiple advanced features based on the group's interests to improve performance. Your grade is composed of both technical and non-technical components, as outlined below.

1 Processor Requirements

1.1 Base Features (required)

1. **A base out-of-order processor.** You will be implementing one of the out-of-order designs covered in class, either P6 or R10K style, although nonstandard designs may be approved. Specifically, your design will need to be able to send instructions to the execution stage in an order other than program order and your report must include a code segment that demonstrates this capability. Your project must support in-order commit or otherwise provide a mechanism for exceptions to be handled correctly (although you only need to demonstrate branch misprediction recovery). **The number of CDBs in your design is limited to the superscalar-ness of your design. E.g. if you are designing a 3-way superscalar design, you may not have more than 3 CDBs.**

2. **Multiple functional units with varying latencies.** You should split the supplied integer ALU into multiple units with different functions and potentially different latencies to improve your cycle time. Most integer operations (other than multiply) should take 1 cycle to execute. Branch target calculations and effective address calculations could also be split into separate units. Use the synthesis tool to guide your decisions. For your multiplier you are to use the one provided in programming assignment 2 although you may change the degree of pipelining as needed.

3. **Instruction and Data cache.** The base memory will have a 100ns latency associated with it. You will be required to build separate instruction and data caches to improve this. The main memory module will be provided, as will a basic I-cache module. **Your instruction and data cache may not be larger than 256 bytes each (so 512 bytes in total).** You are required to use the included `memDP` module as your cache data structure within your I-cache and D-cache.

4. **Branch prediction with address prediction.** You must implement, at a minimum, a branch target buffer and a bimodal branch predictor. More sophisticated predictors may count as advanced feature points.

1.2 Advanced Features (17 points, at least one difficult)

Advanced features are graded out of 17 points (with any over 17 counting as $\frac{1}{3}$). Each group must implement at least one difficult advanced feature and some other advanced features. We suggest that you target earning 16-18 points total. In all cases, *the quality of your feature implementations will play a major role in how we score it.*

Do not be overly aggressive with your advanced features. Many of the features listed below are significantly more challenging than they appear, especially with memory components. In this project, having a correct processor that synthesizes correctly will get a much better grade than an overly complicated one that fails even a few test programs. We recommend iterating on your design; i.e. getting a simpler working processor first and then adding more advanced features as you go.

Some "difficult" and "simple" advanced features are listed below (though you should feel free to suggest others!). Those with more stars (*) are treated as increasing in difficulty. Difficult features are generally worth 6-8 points plus 2 per star (*8-10, **10-12, ***12-15). The simpler features are generally only worth 0.5-3 points. However, these numbers are not absolute and will depend on implementations. Going for multiple high difficulty features has been done in the past, but please talk to us about the risks associated with that. Also, better and/or nonstandard

base feature implementations may count as advanced feature points - talk with staff if you have any ideas here. Textbook chapters (e.g. H&P [N]) are notated as reference material for several of the features, and finding papers on the subjects is recommended. You are also encouraged to do your own research or propose ideas not listed here.

Difficult Advanced Features (estimated: 6-8 points, +2 per *):

- Superscalar execution (2-way, 3-way*, N-way**) (H&P 3.8)
  - i.e. issue, execute, retire width >1. Scalar width is minimum width of entire pipeline
  - Note that you may not include more CDBs than the narrowest part of your design (see above)
- Early branch resolution (before the branch hits the head of the RoB)
- Early tag broadcast
  - Optionally, with speculative scheduling of instructions dependent on load hits
- Speculating on load dependencies and forwarding optimally in nearly all cases (H&P 3.6)
- Multi-path execution on low-confidence branches (this may not help performance much...)

Simpler Advanced Features (estimated: 0.5-3 points):

Some simpler features like prefetching can be fairly trivial, while others can be more complex based on your implementation. Talk to staff to get a good sense of difficulty. The ones with a † tend to give a solid return-on-investment in either performance or debugging.

- Fetch enhancements (H&P 3.9)
  - More sophisticated branch predictors †
  - Return address stack
- Memory hierarchy improvements (H&P 2)
  - Instruction prefetching †
  - Dual-ported, banked L1 data cache (supports two accesses per clock if to independent banks)
  - Associative caches †
  - Non-blocking L1 data cache
  - Victim cache
- Well-made unit tests with good coverage of individual modules.
- Issue memory accesses out-of-order (while still giving the correct results of course!) using a Load-Store Queue (H&P 3.6)
  - There are varying degrees of complexity to which one can implement an LSQ. We recommend being relatively conservative with your LSQ optimizations, at least to start, as this can be a tough component to debug.
- Data forwarding from stores to loads (H&P 3.6)
- Having a really good GUI debugger
  - This entails something beyond just dumping the state of your machine to text files. You should be able to pause, step through cycle-by-cycle, and graphically view the state of various components of your machine.
  - This is only recommended if your group has someone with a strong software background who is comfortable spending a bulk of the project doing software development
  - While this can be extremely difficult, the difficultly is not on the computer architecture side, so the advanced feature points will not be proportional to the effort put in
  - Try not to re-use the vtuber source code since it's not particularly well-designed.

2 Grading

This project is worth 35% of your course grade and will be graded by the weighted average of scores in the following areas:

1. **Implementation of base features** (20%): Did you implement all of the listed base features?

2. **Correctness and testing** (20%): Does your pipeline correctly implement the ISA, and did you convince us of that through your testing methodology?

3. **Performance** (20%): How well does your pipeline perform on the test benchmarks provided (including, but not limited to, the ones used for programming assignment 3)? Performance will be calculated using the CPI derived from the Verilog simulator and the timing reported by the synthesis tool.

    - If your processor does not produce correct synthesis output for a given program, you will get the performance score of the lowest performing processor in the course.

4. **Advanced features** (17%): Points for ambitious designs and interesting implementations. Ideally these points will be in addition to the performance points that these features provide. In the worst case, they are points for trying something bold that didn't quite work out. These are calculated out of 17 points based on what advanced features you implement, see below for details.

    - This category is tied closely with correctness and performance
    - More advanced features help your performance
    - But being too ambitious with your advanced features risks ending up with an incorrect processor
    - Advanced features may significantly increase your clock period which has a large impact on performance

5. **Project Management** (2%): How well did you divide tasks between members of your team? Did you set clear goals, collaborate effectively, and meet the deadlines for your goals? This involves setting weekly milestones to align project sub-groups and bring together everyone in the group

6. **Analysis** (10%): Did you uncover the impact of your features on performance? For example, on a superscalar machine how many instructions do you complete per cycle? What is the prediction accuracy of your branch predictor and/or BTB? How full is your ROB? If you add an interesting advanced feature it would be nice to learn how successful it is with respect to performance. Note that your grade here won't suffer from showing us that something is actually a bad idea. What we want to see is that you can measure how good or bad the idea/feature was. This data will show up in your report.

7. **Documentation** (6%): Did your report describe your design, the motivation for your design decisions, your testing methodology, and your performance evaluation in a readable, concise manner? Although the documentation itself counts for only 6% of the project grade, your scores for other areas will be based on the information you provide in your report. A poorly written report could bring down your scores in all areas.

8. **Milestones** (3%): Did you meet the requirements of the first milestone? Was the module a reasonable one and did it work? Were you prepared for the second and third milestones?

9. **Peer feedback** (2%): Did your group get your CATME peer feedback in on time? Did you take it seriously?

    - There will be approximately 3 CATME assessments throughout the project. In these you will rate the performance of your teammates and leave comments for them that we will release back to you.

3 Project Proposal

A one to two-page project proposal is due **Wednesday 10/1 at 11:59 pm** on Gradescope. The proposal should include:

- Your team number and a name
- A list of group members (with uniqnames / email addresses)
- Base design details (are you planning on implementing a P6-style design? R10K? Something else?)
- Advanced features you are considering
  - Right now, we care mostly about just the difficult advanced features
  - You don't need to fill up all 17 advanced feature points right now since many small features can be added later
  - Keep in mind that memory (LSQ and caches) are much more difficult than you would think, so budget a difficult advanced feature's worth of work for this
- A schedule for which specific goals will be achieved by each of the milestones (e.g., which component will you be implementing for milestone 1)
- An outline of how you will approach managing the tasks for this project. This should include the following:
  - An outline of how you will approach dividing tasks throughout the project (what types of sub-tasks you will have, the number of people on each task, who is responsible for assigning tasks, etc.)
  - Who will be responsible for which specific tasks *prior to the first milestone*
  - A higher level description of who will be responsible for which technical tasks throughout the project. Examples include designing the branch predictor, writing tests for the LSQ, integrating the memory components, etc.
  - A management strategy with specific people assigned to specific managerial roles such as scheduling meetings, taking meeting notes, checking in on others' progress, and (optionally) incorporating fun activities (ex: bringing snacks to meetings)
- A "group charter", outlining your non-technical expectations for one-another, including:
  - How many hours/week are expected per person
  - When during the week each person is typically free and when they are not
  - When the team plans on doing most of its work
  - Notes of each person's strengths and how they would like to grow over the course of the project. This will be very helpful for assigning tasks to the people best suited for them.
  - When and on what the team members will work as a group, and when they will work individually
  - Known conflicts that will take each person away from their work for a prolonged period (off-site interviews, family gatherings, holidays, etc.) or other major class deadlines that will take a person away from the project for more than a day
  - When and where team meetings will be held. In addition to a regular "work" meeting schedule, we recommend scheduling at least one non-work meeting a week to touch base and to interact without formally discussing the project. Holding such meetings over lunch is often a good idea. It can help with communication and morale.
  - *Each person should sign this document (digitally is fine) indicating that they have read and agree with it.*

Your proposal is not a contract; it is likely you will be changing your goals as the semester goes on, but changes should be made with group consensus. The more detail you can give us up front on your plans, the better the feedback and advice you will receive from us during your proposal meeting.

Group members will also periodically fill out CATME peer-evaluation surveys on whether their teammates are participating in the project and acting respectfully towards one-another. This will not be used to "micro-manage" points at the end of the project (all members will usually receive the same grade except in unusual circumstances), but we would like to identify group dynamic issues early on. Please come to the staff early if you have concerns about keeping all group members productive.

4 Milestones and Submission

There will be multiple milestones due for this project where you will submit progress reports and meet with the instructors about your status on the project. The second and third milestones are not hard requirements, but recommended targets for your group to hit.

4.1 Milestone 1

The first milestone is due **Wednesday 10/15**. For this milestone, you are to design a working module and testbench for either the Reservation Station or Reorder Buffer and the interfaces for **at least three** other substantial modules. The goal is that by this milestone you have completed the core of your processor's out-of-order logic.

Your module must have a corresponding testbench, and be able to generate correct output in both simulation and synthesis. The three module interfaces don't need to be implemented yet, but you must at least have an idea of how they would. This will help you divide the responsibilities of implementing these modules' functionality and simplify integration later. An example (incomplete) interface is below:

```
module BranchPredictor (
    input        clock,
    input        reset,
    input  ADDR  PC,
    ...
    output logic taken,
    ...
);
    // TODO: Add implementation here :)
endmodule
```

For this milestone, create a specific branch of your git repository with the working module and 3 module interfaces, and specify this branch in your one-page report which you will turn in on Gradescope. We will grade your working module for completeness (does it do everything such a module should do) and style. We will grade your testbench for coverage and style through injecting bugs in your module and testing if your testbench detects them.

Your (brief) report should indicate your progress to date, progress relative to the original schedule (referencing the task breakdown you specified in your proposal), how you will divide tasks going into the second milestone, a reflection on how your project management is going, and any changes in the scope or direction of the project relative to the original proposal.

4.2 Milestone 2

The second milestone is due **Friday 10/31**. Another brief progress report is required, and your basic components (excluding instruction cache) should be integrated into a functional pipeline such that most non-memory operations can be correctly fetched, decoded, executed, and committed. You should be able to run the program `mult_no_lsq.s` with the output in the `.wb` and `.out` files being correct (matching the output from your project 3 processor). If this is done, we suggest working on getting `btest1` and `btest2` programs functioning correctly, as these stress the control aspects of your processor (branch predictor, instruction squashing, etc.).

Your report should also include an update on how your project management strategies are working, and if you plan on making any changes to your management structure. Please include your progress on your tasks listed in your milestone 1 report, including when each task you originally planned for was completed. Also include your planned division of tasks going into the third milestone and final submission.

4.3 Milestone 3

The third milestone is due **Wednesday 11/19**. Another brief progress report is required though there will be no required group meeting. *Ideally, at this point you have a fully functional processor that can run all programs correctly*, and you are now working on performance optimizations (reducing your critical path, optimizing performance, etc.). This includes your project being synthesizable, with multiple programs finishing correctly under a synthesized processor.

If you don't have a working processor, your remaining work should be in the memory subsystem at this point. If you are still working on getting your data cache and LSQ working at this point, consider dropping some complexity in order to have a correct processor by the end of the project.

Note that this milestone is barely over two weeks before the final deadline, and Thanksgiving break falls during this timeframe as well. Keep this in mind as you plan out your timeline so you aren't left with too much work in the last week of the project.

4.4 Final Submission

Your final project Verilog code (to be submitted electronically for testing) is due **Saturday 12/6 at 11:59pm**. The written project report is due **Monday 12/8 by 11:59pm**. The project report should be 5-10 pages in length and include an introduction and details on the design, implementation, testing, and evaluation (analysis) of your pipeline, including specific discussion and analysis of any advanced features.

There will also be brief, relatively informal oral presentations at the end of the semester. More details about these will be given later, but you can at least expect some to be during class on Tuesday 12/8.

5 Final Suggestions

Here are some final suggestions for getting started:

- **Code your first module as a group.** It will help immensely with getting everyone on the same page and working out coding standards as a team. For larger groups this can feel like a waste of time, but it will save you much more time later.
- **Divide the work into small sub-tasks with clear deadlines**. Each task should ideally go to 1-2 people; more than that adds too much communication overhead. Dividing work allows for good parallelization, and deadlines allow you to integrate work efficiently.
- **Use an iterative design approach**. Set weekly internal milestones. Good designs are the result of iteration and continual refinement.
- **We will provide a starting point for a couple of modules.** While you may or may not directly use these modules, they provide some helpful guidance for your project design. This information will be posted soon after P3 is turned in. You may reuse freely from P3 (the decoder is a good thing to not have to re-write).
- **Generate your own test programs**. Although we give you some sample test programs, generating your own allows you to create targeted tests that provide more comprehensive testing of your processor. You can get some extra tests for "free" by changing the GCC compiler optimization flags in your Makefile.
- **Read the updated README and use the Makefile.** The README for project 4 has been updated with instructions for using the new module testbench system in the Makefile. This will help with managing all the different modules and testbenches you will be writing.