# CPRE 581 Final Project Topic Selection

**Date:** 10/14/2023

**Member List:**
- Jackson Hafele, jmhafele@iastate.edu
- Gregory Ling, gling@iastate.edu
- Thomas Gaul, tvgaul@iastate.edu

## 1. Summary

For this project, we are proposing to simulate and create RTL designs for a set of different branch predictors using Gem5 and an RTL implementation. We are planning to build off of the different branch prediction units and a branch target buffer that has been reviewed in class and homeworks, and implement them in a 5-stage MIPS pipelined architecture that we created in CPRE 381. In Gem5, we will be able to use cycle-level simulators to gain insights on prediction accuracy for simple pipelined processor designs. We already have a synthesized 5-stage pipeline, and can gain insight into timing requirements and power estimations using synthesis tools. As an added bonus, we can attempt to synthesize our design onto an FPGA using our previous extra credit project from CPRE 381.

## 2. Related/Base Work

https://dl.acm.org/doi/abs/10.1145/165123.165161
- This paper compares multiple configurations of 2 level branch predictors. By using multiple combinations of global, per-set, and per-address prediction schemes, prediction accuracies were able to be compared for all possible 9 combinations of predictions. We could use this paper as an example for background context on each type of predictor, and how we can display our prediction results from our Gem5 simulations.

https://dl.acm.org/doi/abs/10.5555/545215.545249
- This paper looks at design tradeoffs on the Alpha EV8 processor. It looks into various constraints faced by the team while they were defining the branch predictor. It uses a global history prediction scheme 2Bc-gskew. With their design, they hit between 10 and 1 misprediction per 1000 instructions depending on the workload.

https://ieeexplore.ieee.org/document/6062308
- This paper describes branch hint instructions attempted in an IBM processor. Some limitations they ran into were a minimum distance between the hint instructions and the real branch instruction which required them to use NOPs to fill the gap, and the need to restructure nested loops to allow for enough space to fit the branch hint instructions before the actual branches. Even with these limitations, they appear to have gotten some improvement from this design.

## 3. Proposed Work/Idea

Possible Branch Predictors:

- Null predictor
- Random predictor
- N-bit local predictor
- Global predictor
- Gshare
- Tournament

We would also like to try adding a hint instruction to "prime" the branch predictor to see how this compares with the branch predictors above. This hint instruction would evaluate a condition as a branch instruction, but instead of branching immediately, inform the branch target buffer that there will be a branch at that PC, but no branch will actually be present. This would give the added benefit of allowing the compiler greater flexibility than a branch delay slot, but also give a warning to the processor so it knows the destination of a conditional branch earlier in the program execution.

```
  pbnz r0, branch (#8), target (#12)
  movi r1, #10
  addi r2, r1, #30
branch:
  sub r2, r2, r0
target:
  movi r1, r2
```

```
  movi r1, #10
  addi r2, r1, #30
  beq r0, r1, target
  sub r2, r2, r0
target:
  movi r1, r2
```

## 4. Likely Evals/Tools Used

For simulation purposes, we plan to use gem5 to model different branch predictors with cycle-level simulations to compare their performance, prediction accuracy, and complexity of implementation. We were considering using either a generic superscalar pipelined processor, or a stripped down configuration to remodel our MIPS processor in gem5, to help set a baseline for our FPGA configuration. For the RTL implementation, we plan to use ModelSim to simulate our waveforms and use Verilog to create our design modules and testbenches. We will synthesize our project using Quartus Prime, and will use one of the ETG's DE2 FPGA Development boards to synthesize our MIPS FPGA, which was completed in CPRE 381. We will build on this by adding and comparing different branch predictor units after each module is verified.