MP4.3 Report

Neo Vasudeva, Naveen Nathan, Anchit Rao

Progress Report

Perceptron Branch Prediction (*8 points*)

Implemented Functionalities: Branch prediction to preload instructions and operations based on patterns in current program instructions. Branch prediction hovers around 70% for the competition code test cases and 90% for CP3. Our implementation has a global branch history length of 12 and a table of 32 perceptrons.

Testing Strategies: Tested using competition code 2 (large number of branches taken) as well as the other competition codes. However, the branch predictors accuracy is the highest on the checkpoint 3 test cases when compared with the competition codes. Small testbenches were also written during the implementation of the perceptrons to test parts as written.

4-Way Set Associative BTB (*3 points*)

Implemented Functionalities: BTB stores the PC address along with the predicted branch target. Since it is a 4-way set associative BTB, it runs faster compared to a direct mapped BTB with the same number of entries. Our implementation uses 8 sets.

Testing Strategies: Tested using competition code 2 (large number of branches taken) as well as the other competition codes. However, the branch predictors accuracy is the highest on the checkpoint 3 test cases when compared with the competition codes. Currently, our branch prediction hovers around 70% for the competition code test cases. Small testbenches were also written during the implementation of the BTB to test parts as written.

8-Way Set Associative L2 Cache (*5 points - 2: L2 | 3: 8-way*)

Implemented Functionalities: The 8-way set associative L2 cache is a faster cache shared between ICACHE and DCACHE to improve storage speeds and hit accuracy.

Testing Strategies: First, the mp3 cache code was cleaned up and tested with previously created assembly code to test evictions. Then a single node pLRU tree was created in place of the LRU array in the mp3 cache and tested against the same assembly codes. Following this, the cache was expanded to 4-way associative and the pLRU module was modified as such. New assembly code was written to test evictions on this 4-way associative cache. Once this worked, we again expanded to 8-way associative and further added new assembly code to test evictions on the 8-way associative cache. At this point, we knew the cache functionality was solid, so all we had to do was add it into the cache hierarchy. Once this was done, we confirmed functionality with performance counters (counting the number of misses vs the total number of reads and writes in L2) in mp4-cp3 and all of the competition codes.

Explanation of Inclusive/Exclusive (*1 point*)

Inclusive L2 caches contain all the information within the L1 class along with additional. Conversely, in exclusive caches, data is guaranteed to be in at least one of the two caches. Furthermore, inclusive caches are lower in performance when compared to exclusive caches; however, exclusive caches are more efficient, with regards to performance, but come with a more complex cache coherence policy. We chose the inclusive design as we believed that the additional complexity would not have been worth the minimal change in performance in our use case.

Roadmap

Performance Improvements - *Neo, Naveen*

We will work together to drive up the f_max as much as possible. Currently our f_max is 63 MHz. We will also try to improve branch prediction accuracy.

Report - *Anchit*

We are planning on adding our design diagrams from our original blueprints as well as any optimizations we have added. Furthermore, we will include important parts from our progress reports through the different checkpoints. All other information will be added as per the rubric.

Presentation - *Neo, Naveen, Anchit*

Our presentation will be worked on by all of us together. We are planning on adding our design diagrams from our original blueprints as well as any optimizations we have added. Furthermore, we will include important parts from our progress reports through the different checkpoints.