

ECE411 MP3 Final CP Report

Group: ZerotoOne

Progress Report:

By the final checkpoint, we successfully rooted out the bugs in our system to correctly run the final test code, as well as implemented advanced features that improves the performance of the system. We implemented the following advanced features:

1. A tournament branch predictor with both local history branch predictor and 2-level global history branch predictor. A four state saturation counter keeps track of the prediction accuracy of the predictors and output the result of the better one.
2. A branch target buffer storing the target to jump to for branching and jumping instructions. It is stored as a fully associative array and has a round robin replacement policy. The branch instructions and jump instructions share the same buffer.
3. A dynamic hardware data prefetch-unit that has a reference prediction table(RPT) that stores memory access info for needed PC. For each PC mapped memory access info, our(RPT) stores previous-access-address, stride, and state. These information are used to possibly generate an outstanding request address whenever a new PC is updated.

Contribution:

Haiyang Zhang: implementation of tournament predictor, BTB, and debugging

Yuan Ma: implementation of the mem data prefetcher and debugging

Roadmap:

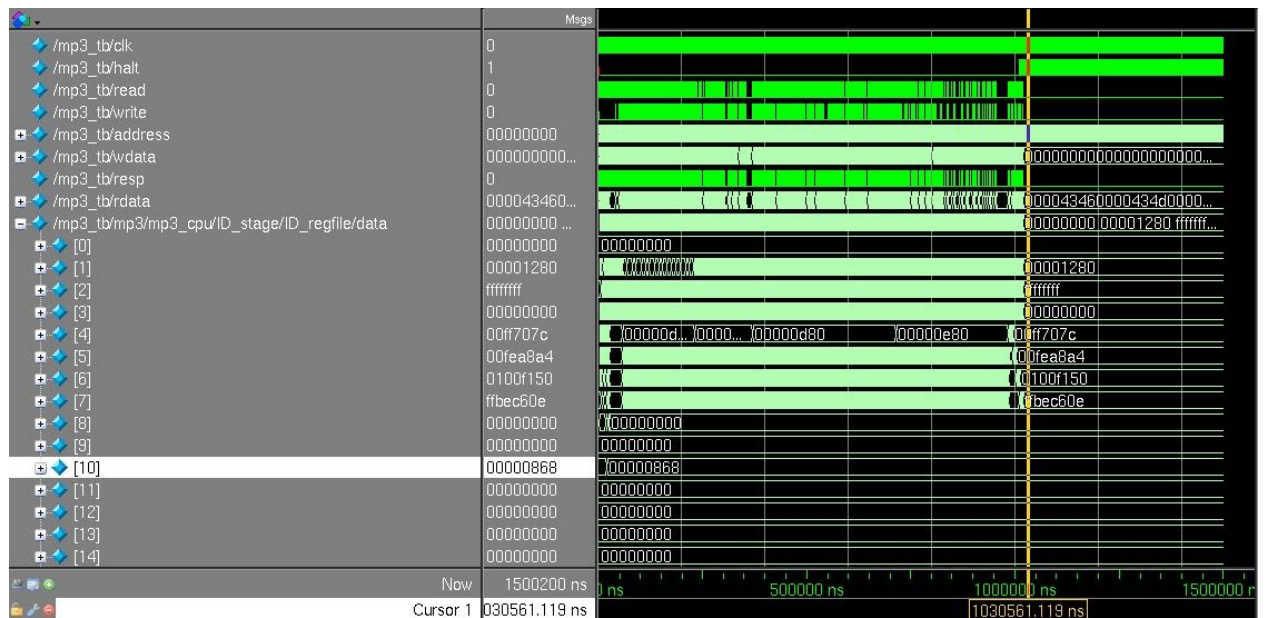
By the end of the competition demo, we expect to :

1. Continue to improve the performance by pipelining the L2 cache to improve the max frequency.
2. Continue to explore the potential of the advanced features that we implemented, including finding the best size for the BTB or making it set associative, adding support for jalr in the branch predictor

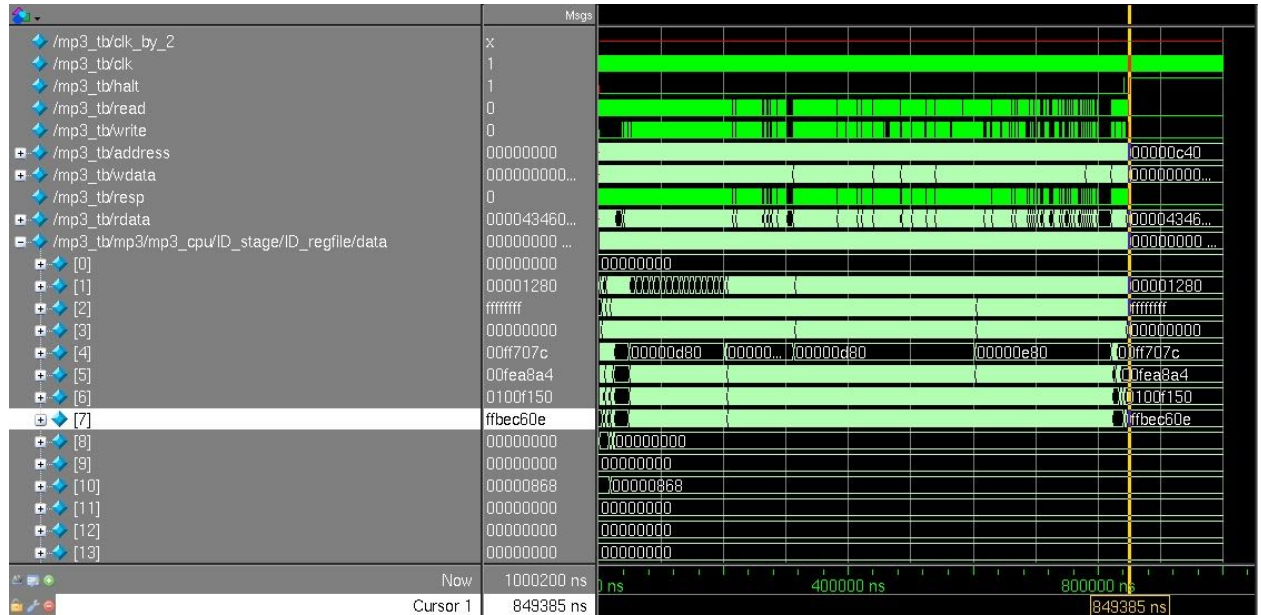
Testing and Performance:

1. BTB and tournament buffer: The best test program is one with a lot of branching instructions , so we mainly use the given cp3-final test code for the testing. We believe it is sufficient because it has sufficient amount of branches to fill the BTB and also covers all kinds of branching and jumping instructions. By this code we were able to discover a problem associated with jalr instructions where the jump target is not constant.

The way we test its functionality is mainly by comparing how much time it saves to execute a long program full of branches. The picture below shows how much time it takes for the program to finish executing mp3-final with only static prediction.

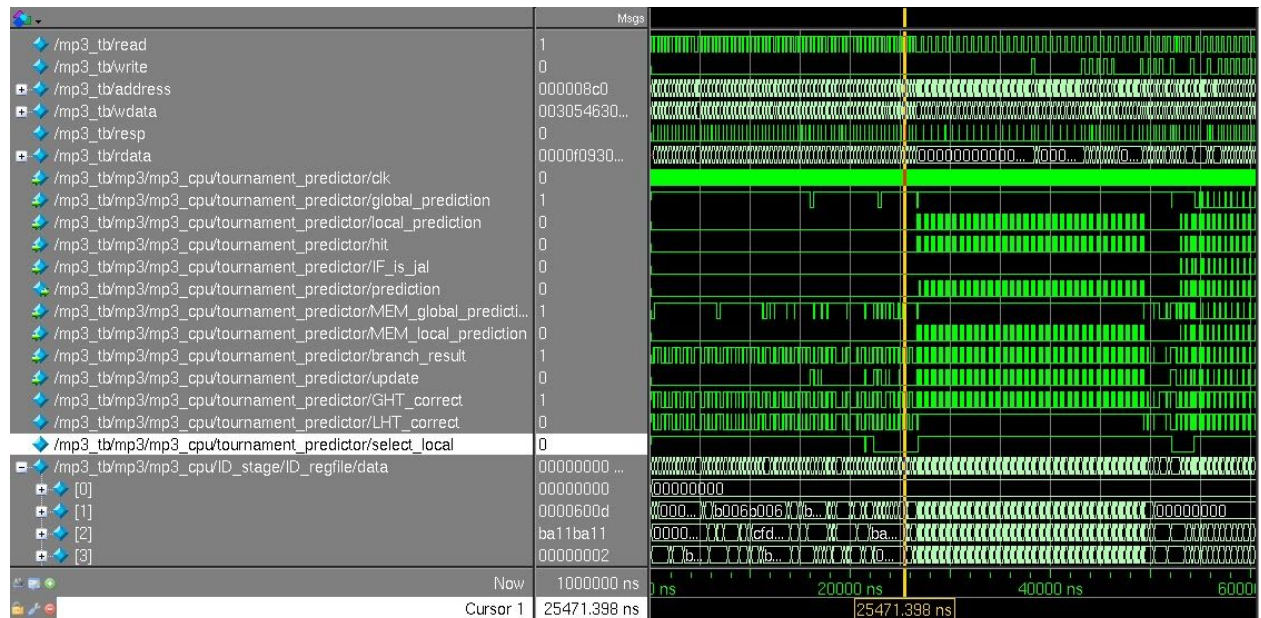


By the end of the program, there are around 0x31e8 branches/jumps and 0x249b misses, which accounts for 73% of the branches. And this picture shows how the performance improves with the tournament predictor



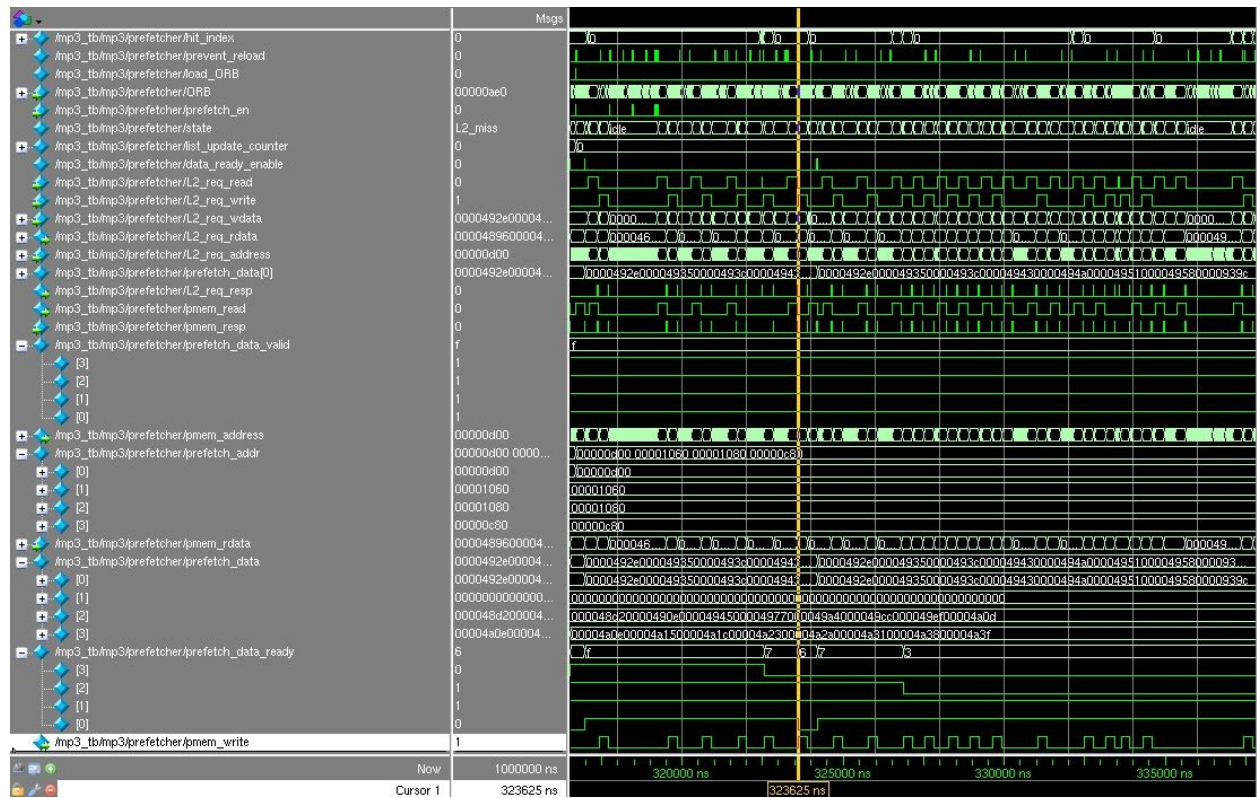
The execution time shortened from roughly 1030,000 ns to 850,000 ns, which is almost 20 % performance increase. According to the reading of performance registers, by the end of the program, there are 0x31e8 branches and 0xcdf misses, which accounts for 26% of the branches.

The following picture is a closer look at the tournament predictor. It is clear that the predictor is making a choice from the local and global predictor. The local one is chosen most of the time but the global one is also used sometimes.

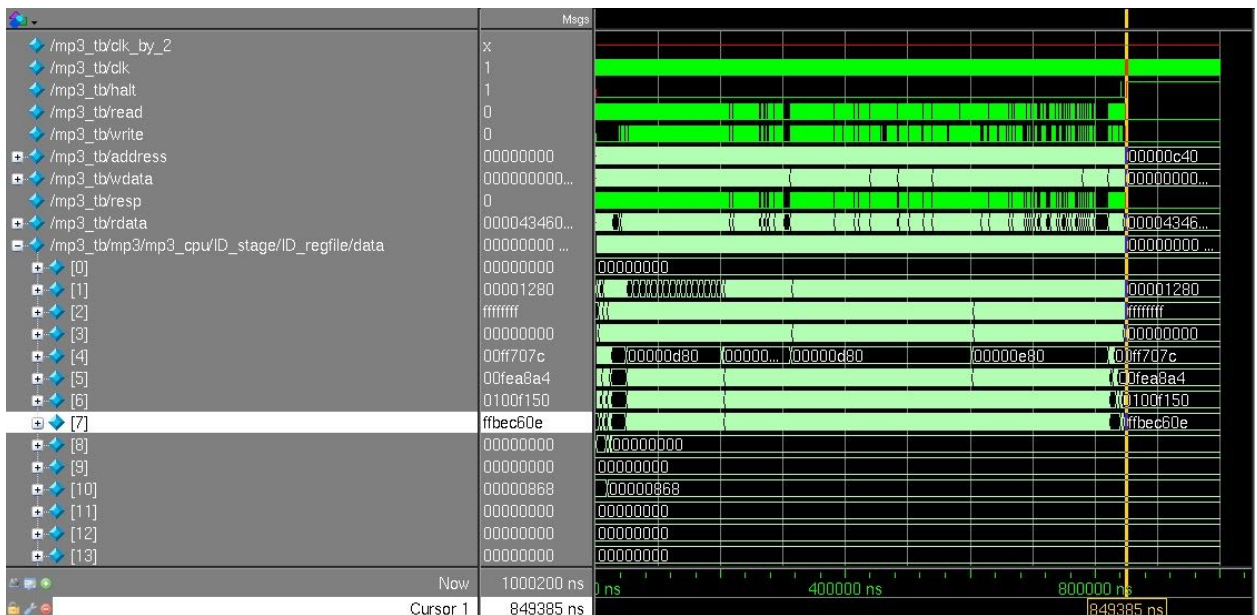


2. Dynamic Stride Hardware Prefetcher: we use the mp3-final.s as our test-code prototype for its significant number of L2 miss and replacement. The corner test includes the scenario that multiple PC access the same address frequently. This tests whether the prefetcher is able to send only one predicted address to the PMEM. Another corner test includes read the memory that is written back immediately. This tests whether the prefetcher is able to detect the dirty data that has been accessed.

Below is a corner situation that show an immediate pmem write instruction that makes the prefetcher preloaded data invalid.



We compare the performance before and after adding the prefetch-unit. The performance before adding the prefetch-unit is around 849,300.



After adding the prefetching unit, we found a minor improvement in runtime from 849,300 to 847,200, nearly 0.5 percent. We think the reason is that we generate the potential prefetchable data in IF stage, only 3 clock cycle ahead and a cache miss immediately block prefetching process. All these reasons significantly reduces the hit rate in the prefetched data.

