Richard Wu | wuricha8 | 1008078296
Christian Vedtofte | vedtofte | 1012872125

# ECE552 Lab 2 Report

## Microbenchmark for 2-Level Predictor

Our microbenchmark, mb.c, has three conditional branches: (i) a loop (10,000,000 iterations) condition that can be considered taken every time; (ii) a branch that is not-taken every 10 iterations; and (iii) a branch that alternates between taken and not-taken. For (i), the history register will be 6-bits of 1's (i.e., taken) and will predict taken every time, so there will be no mispredictions here. For (ii), since 6 bits of history cannot capture 10 instances of the branch outcome, the 2-level predictor will mispredict each time the branch is not-taken, so 1/10 of the time. For (iii), the history registers will either be [1 0 1 0 1 0] or [0 1 0 1 0 1] and will predict correctly, so no mispredictions here. Thus we expect to get ~1,000,000 mispredictions. There are 13 assembly instructions per loop, and, taking into account skipped instructions from taken branches, we expect the MPKI for the 2-level predictor to be about (1)/(9*5+10*4+11*1) * 1000 ~= 10.41. The resulting MPKI was 10.424, which is very close. Our compilation command is:

```
gcc -O0 mb.c -o mb
```

## Performance Results of 3 Predictors with Given Benchmarks

| BM | 2bitsat (mispredicted branches) | 2bitsat MPKI | 2-level (mispredicted branches) | 2-level MPKI | Opened (mispredicted branches) | Opened MPKI |
|---|---|---|---|---|---|---|
| astar | 3695830 | 24.639 | 1785464 | 11.903 | 424020 | 2.827 |
| bwaves | 1182969 | 7.886 | 1071909 | 7.146 | 176960 | 1.180 |
| bzip2 | 1224967 | 8.166 | 1297677 | 8.651 | 1244818 | 8.299 |
| gcc | 3161868 | 21.079 | 2223671 | 14.824 | 113144 | 0.754 |
| gromacs | 1363248 | 9.088 | 1122586 | 7.484 | 800292 | 5.335 |
| hmmer | 2035080 | 13.567 | 2230774 | 14.872 | 1934810 | 12.899 |
| mcf | 3657986 | 24.387 | 2024172 | 13.494 | 1521035 | 10.140 |
| soplex | 1065988 | 7.107 | 1022869 | 6.819 | 614268 | 4.095 |
| **AVG** | **2173492** | **14.48988** | **1597390.25** | **10.649125** | **853668.375** | **5.691125** |

## Open Ended Predictor Implementation

For our open-ended predictor, we implemented a simplified version of the TAGE predictor[1] with optimizations targeting the given suite of benchmarks. We use 9 T-blocks of {128, 512, 512, 512, 1024, 1024, 1024, 1024, 1024} entries respectively. Entries of block T0 (base predictor) contain a 4-bit bimodal counter. Entries in the other 8 blocks (T1-T8) contain: 3-bit bimodal counter, 2-bit useful counter, and 11-bit tags. The global history register (GHR) is 256 bits long. The

[1] André Seznec, Pierre Michaud. A case for (partially) tagged geometric history length branch prediction. The Journal of Instruction-Level Parallelism, 2006, 8, pp.23. ffhal-03408381

Richard Wu | wuricha8 | 1008078296
Christian Vedtofte | vedtofte | 1012872125

T-blocks use {0, 2, 4, 8, 16, 32, 64, 128, 256} bits of the GHR for hashing, respectively (T0 has a simple PC only indexing scheme). Bimodal counters are initialized to 3, useful counters and tags are initialized to 0. The total storage is: (GHR bits) + (T0 entries) * (4 bits) + (512 entries) * (3+2+11 bits) * (3 blocks) + (1024 entries) * (3+2+11) * (5 blocks) = 256 + 128*4 + 512*16*3 + 1024*16*5 = 107,264 bits = **104.75Kbits**. This is within the 128Kbits limit.

Computing the prediction first requires 2 separate hashes of the PC plus GHR bits for each T block. Hash1 is used to index the entry in the T block and hash2 is used to check for a tag hit. We use the prediction of the bimodal counter belonging to the tag-hitting T block using the longest GHR bits. If none of T1-T8 have a tag match, we use the base prediction in T0. Upon resolving the branch, the bimodal counter used for the prediction is updated accordingly. If the prediction was correct, the useful counter of the used entry is incremented. Otherwise, we search for an entry in a T block using greater history with a useful counter of 0 and initialize that entry, setting the tag to the appropriate hash2 value. If we can't allocate, we decrement the useful counter. Both our hash functions involve folding PC and GHR bits and taking the XOR[2]. Hash1 and hash2 differ by taking different lengths to fold.

**CACTI Results for 2-Level and Opened Predictors**

| | **2-Level (PureRAM)** | **Opened (GHR, T0)** | **Opened (T1-T8)** | **Opened (total)** |
|---|---|---|---|---|
| **Area (mm^2)** | 0.001935555 | 0.000385911 | 0.020176442 | 0.020562353 |
| **Access Latency (ns)** | 0.164342 | 0.150048 | 0.262473 | 0.412521 |
| **Leakage Power (mW)** | 0.366395 | 0.0649702 | 4.6092 | 4.6741702 |

We used the pureRAM config for the 2-level predictor; the only parameter that changed was -size = 1024. The BHT table has 512 entries of 6-bit history registers, equaling 512 bytes; the PHT table consists of 8 tables of 64 2-bit entries, also equaling 512 bytes, thus total 1024. For our open-ended predictor, we made one config (open-ended-bpred-1.cfg) using the pureRAM structure to model the -size = 160 bytes of the GHR and T0 (see file comments for more). For the rest of the T blocks, we also used a pureRAM structure, for a total of -size = 13312 bytes, with -block size = 2 (since entries are 16 bits = 2 bytes) and thus -output/input bus width = 16 (see file comments for more). Since both configs are pureRAM structures, the total values can be a simple sum of both (see table above).

**Statement of Work**

Richard and Christian both worked on the 2-bit saturating predictor and 2-level predictor. We both did research on TAGE for our open-ended predictor. Richard implemented the prediction computation and Christian helped with the hash function and CACTI configurations for both 2-level and opened predictors.

[2] P. Michaud. A ppm-like, tag-based predictor. Journal of Instruction Level Parallelism (http://www.jilp.org/vol7), April 2005.