ECE552H1F Lab2 Report

Jianwei Sun 1000009821
Yi Fan Zhang 1000029284

Microbenchmarking

As there are 6 history registers linked to a table of bimodal counters, the micro-benchmark explores three different branch cases; (a) where there is a branch that changes direction every 8 bits (outside the history), (b) where the branch direction toggles, and (c) where the branch direction stays the same.
The first if statement looks at case a; because this requires a BHR of at least 8 bits, we would expect a mis-prediction at every 8th iteration of the loop.
The 2nd if statement looks at the toggling of branches; since this prediction changes every time, we would expect the bimodal bits toggle with the previous branch prediction; this means that it would give a wrong prediction every time.
Finally, the main for loop is a branch direction that doesn't change; because of the bimodal bits keep track of history, this should always be correctly predicted.
As the BHT and PHTs are indexed by PC, and the PC locations of the branches don't change, we would expect the updates to be mostly in the same location after the BHR settles.
From the assembly code, there are 0x400278 – 0x400238 = 20 instruction per iteration. Therefore, MPKI = (8+1)*(1000/(20*8) = 56.25. The microbenchmark is simulated against the 2-level predictor giving a MPKI = 54.618, which is mostly consistent with calculations.

Branch-Prediction Implementation

Performance Results of Three Branch Predictors Tested with Eight Programs

	2bitsat Mispredicted Branches	2bitsat MPKI	2levelpred Mispredicted Branches	2levelpred MPKI	Openend Mispredicted Branches	Openend MPKI
astar	3697270	24.648	3709167	24.728	422846	2.819
bzip2	1227534	8.184	1282347	8.549	1253416	8.356
gromacs	1363971	9.093	1401146	9.341	832419	5.549
mcf	3658361	24.389	4998917	33.326	1493033	9.954
bwaves	1186188	7.908	1203008	8.020	162230	1.082
gcc	3179496	21.197	3320539	22.137	87776	0.585
hmmer	2035534	13.570	2036837	13.579	1935480	12.903
soplex	1066980	7.113	1235988	8.240	6.12475	4.083

We implemented a TAGE predictor as the open-ended portion of this lab. We have chosen a total of nine Tblocks (include the default), with 3-bit bimodal counter, 2-bit useful counter, and 11-bit tag sizes. The global history register (GHR) was chosen to be 512 bits long. We represented these data structures as a series of arrays, as that was the most logical choice. We also implemented two hash functions for addressing and tag matching. Our implementation of getting a prediction involved checking for the longest-history Tblock for a matching tag, and looking to the next longest-history until a tag is found. If none is found, the default T0 prediction is used. Upon evaluation of the branch, update occurs by incrementing the corresponding Tblock's useful bit if the prediction was successful. Otherwise, a new entry would be allocated in the adjacent Tblock of longer history length. During allocation, if the

adjacent Tblock's usability bit is 0, we can allocate right over it. If the usability bit is not zero, we decrement it and continue our search in the next Tblock. The corresponding bimodal counters are updated to reflect the branch resolution as well as the GHR updated.

Since the GHR and T0 block are not content addressed, they were represented as a pureRAM type of device. The T0 block has 512 entries, each a 3-bit bimodal counter, which resulted in a total size of 512 bytes for the CACTI configuration for the T0 block. Since the GHR is 512 bits long, it was represented as 64 bytes, which resulted in a pureRAM size of 64 bytes. Combining these two structures resulted in a total size of 576 bytes, with associativity of 1. On the other hand, the blocks T1 to T8 were represented as cache structures, due to their content-addressability using the tag bits. The tag width was chosen as 11 bits as a balance between overall performance and physical size, and is reflected in the CACTI configuration file parameter for tag width. The 11-bit tag size with the 3-bit bimodal counter and 2-bit useful counter results in 16-bits total size per entry. In the CACTI configuration file, the block size was selected to be 2 bytes to reflect this entry size. We opted for variable sized Tblocks as a means to improve performance, and settled on the following sizes (number of entries) for blocks T1 through T8, respectively: 896, 1024, 1024, 1024, 1024, 1024, 1024, 1024. The total size of this structure is then 16128 bytes, which is reflected in the CACTI configuration file. T1 was selected to have 896 entries to make full utilization of all available 128kbits of storage.

Based on the specified configuration parameters above, the open-ended predictor results in the following physical simulated results.

CACTI Simulated Results for an Open-Ended TAGE Predictor and a Two-Level Predictor

	PureRAM Structures in open-ended	Cache Structures in open-ended	Two-Level Predictor
Area (mm^2)	0.0011600314	0.0220456899	0.001935555
Access Latency (ns)	0.167106	0.24615	0.164342
Leakage Power (mW)	0.217019	5.13948	0.366395

The CACTI parameters used for the two-level predictor involved the pureRAM configuration. The two-level predictor consisted of a table of branch history registers, 6 bit entries, for a total of 512 entries. The second level consisted of 8 tables, each of size 64 entries, of 2-bit bimodal counters. Since the smallest unit that can be represented in CACTI for pureRAM is a byte, these tables correspond to a total of (512 + 8*64) bytes, which was set as the configuration parameter for the two-level predictor in the CACTI simulation.

We, Danny and Jianwei, completed equal portions of this lab and all its components. We worked together on creating the 2-bitsat predictor as well as the two-level predictor. We both spent time researching different aspects for our open-ended predictor: Danny researching the hash function implementations, and Jianwei implementing and debugging the Tblocks. Together, we solved all the problems with our initial attempts, and optimized our open-ended branch predictor to compete competitively. Danny placed emphases on completing the microbenchmark to a high quality, during which Jianwei completed the CACTI configuration parameters, and collected statistical data on the performance of the various predictors and physical implementation details.