ECE552 Lab 2 Report

Yang Li (1002066792) Hao Liu (1002614487)
October 18, 2019

1 Microbenchmark

For the microbenchmark, we made a loop that iterates itself with counter *i* increases it self from 1 to 100000, while a possible branch will be taken depending on whether the counter is exactly divisible by *LOOP*.

```
#define LOOP 8
int main() {
    int count = 0;
    for (int i = 1; i < 100001; i++) {
        if (i % LOOP) {
            count++;
        }
    }
}
```

With LOOP equals to 7, 6-bit history memory can memorize the pattern since the branch is taken every 7 loops, starting prediction counter from *weak-not-taken*, we expect there to be only 1 miss for LOOP = 7. However, in case when LOOP equals to 8, history register lacks 1 bit to fully capture the history pattern (Taken every 8 loops) on when the branch will be taken, we therefore expecting all the branch predictions to fail, that is 100000/8 = 12500.

LOOP value	Predicted # Mis-prediction	Actual # Mis-prediction (with overhead)	MPKI
7	1	1300	0.572
8	12500	13799	15.800

Eliminating initialization overhead, 13799-1300 = 12500, we can prove the actual result is correct.

2 Results

To ensure consistency, we select a random seed to be fixed integer.

Benchmark	2-bit sat		2 level		openend	
	Misprediction	MKPI	Misprediction	MKPI	Misprediction	MKPI
astar	3695830	24.639	1785464	11.903	421988	2.813
bwaves	1182969	7.866	1071909	7.146	119668	0.798
bzip2	1224967	8.166	1297677	8.651	1240989	8.273
gcc	3161868	21.079	2223671	14.824	93961	0.626
gromacs	1363248	9.088	1122586	7.484	861341	5.742
hmmer	2035080	13.567	2230774	14.872	1996075	13.307
mcf	3657986	24.387	2024172	13.494	1441060	9.607
soplex	1065988	7.107	1022869	6.819	603068	4.020

3 Open-ended Branch Predictor

3.1 Implementation

The open-ended branch predictor for this lab was implemented based on the idea of TAGE predictor.

The TAGE predictor features a base predictor and a set of tagged predictor components. These tagged componenets are indexed using history lengths $L(i)$ that form a geomteric series: $L(i) = (int)(\alpha^{i-1} * L(1) + 0.5)$. Each component contains different number of entries, and each entries consists of a counter *pred*, a *tag* and a useful bit *u*. The index and tag values of the predictor components are calculated through respective hash functions.

At prediction time, the base predictor T0 provides a default prediction when there is no tag hit. When there is a tag hit, the prediction is provided by the hitting tagged predictor component that uses the longest history. We also utilized an *altpred* which provides a "second-best" prediction and a *use_alt_on_na* counter, and these will be evaluated by comparing with the prime prediction result later.

The brief updating algorithm of TAGE is as follows:

- If the alternated prediction is different from the final prediction, update the useful counter *u*
- Update the sat counter *pred* of provider according to the actual branch direction.
- If the overall prediction is wrong, try to allocate one tagged entry to a vacancy on table with $u = 0$ and longer histories. If no vacancy was found, decrements every u counters of the tables with longer histories.
- Reset LSB and MSB of all *u* counter every 256K branches.

3.2 Storage Calculation

For this lab, we use 10 TAGE predictors and 1 base predictor. The base predictor has 8K 2-bit entries, and the TAGE predictors have $\{2K, 2K, 1K, 512, 256, 256, 256, 256, 256, 256\}$ entries with $\{12, 10, 10, 9, 9, 9, 8, 8, 8, 7\}$ respective tag widths. We also stores global history record for 512 bits. So the overall storage bits are calculated as follows: $8K*2+2K*17+2K*15+1K*15+512*14+256*(14+14+13+13+13+12)+512 = 122.25Kbits$

4 CACTI Report

Predictor	Access Time (ns)	Dynamic read energy/access (nJ)	Leakage read/ writepower (mW)	Area (mm^2)
2 level	0.283226	4.24323e-4	0.2305923	2.07043e-3
openend	0.265905	6.0507336e-3	6.899256	0.03313841

2-level predictor features two prediction table (banks) one has 512 6-bit entry and the other has 64 2-bit entries (2level-bpred-1.cfg & 2level-bpred-2.cfg), representing the *size* to be 512 and 64, *tag size* to be 6 and 2 respectively. In addition for 2-level predictor, each prediction requires access to both table in sequence, therefore the access time should be accumulated.

For the open-ended predictor (TAGE), it has 8 distinctly configured table with overall 11 tables. *Size* and *tag size* have been set accordingly based on the geometric series determined on part 3. When making a prediction, the access to all prediction tables will happen in parallel, therefore we can take the longest access time across all tables as our access time. Correspondence of cfg files to distinct tables is shown as below.

Open-ended cfg suffix	-1 (Base)	-2	-3	-4	-5	-6	-7	-8
Total tag size	2	17	15	15	14	14	13	12
# Entries	2^{13}	2^{11}	2^{11}	2^{10}	2^9	2^8	2^8	2^8
# IdenTables	1	1	1	1	2	3	1	1

5 Statement of Work

Both student completes the equal amount of work for this lab.