Incorporating Pattern Prediction Technique for Energy Efficient Filter Cache Design

KUGAN VIVEKANANDARAJAH, THAMBIPILLAI SRIKANTHAN, SAURAV BHATTACHARYA, PRASANNA VENKATESH KANNAN*

Centre for High Performance Embedded Systems
Nanyang Technological University
SINGAPORE.

*University of California,
Santa Barbara
USA.

URL: http://www.chipes.ntu.edu.sg

Abstract: - A filter cache is proposed at a higher level than the L1 (main) cache in the memory hierarchy and is much smaller. The typical size of filter cache is of the order of 512 Bytes. Prediction algorithms popularly based upon the Next Fetch Prediction Table (NFPT) helps making the choice between the filter cache and the main cache. In this paper we introduce a new prediction mechanism for predicting filter cache access, which relies on the hit or miss pattern of the instruction access stream over the past filter cache lines accesses. Unlike NFPT, which makes predominantly incorrect miss-predictions, the proposed Pattern Table based approach reduces this miss prediction. Predominantly correct prediction achieves efficient cache access and eliminates cache-miss penalties. Our extensive simulations across a wide range of benchmark applications illustrates that the new prediction scheme is efficient as it results in improved prediction up to 99.99%. Moreover, it reduces energy consumption of the filter cache by as much as 25% compared to NFPT based approaches. Moreover, the technique implemented is elegant in the form of hardware implementation as it consists only of a shift register and a Look up Table (LUT) and is hence area and energy efficient in contrast to the published prediction techniques.

Key-Words: - Cache, Filter-Cache, Energy Efficient Cache, Pattern Prediction, NFPT, Hit-Miss Pattern

1. INTRODUCTION

With high clock speed and smaller feature size based CMOS design, the power consumed in the embedded microprocessor has been increasing with every generation of microprocessors (Figure 1). Due to this, architecture level low-power techniques have been active areas for research in recent times.

Instruction caches often account for more than 50% of die area and more than 40% power consumption in high-end embedded processors [1]. Though future process techniques can be expected to facilitate bigger instruction caches to be fabricated on chip, power and access time will continue to remain limiting factors. This highlights the need for a memory hierarchy design that maximizes the instruction-hit rate, while reducing access time and energy. Filter cache [1] had been proposed, to achieve 58% power saving at the cost of 21% performance loss. Subsequently Next Fetch Prediction Table (NFPT) [2] based prediction was introduced for predictive access of the filter cache, rather than a direct hierarchical access, thereby limiting performance degradation to 1.3%, at the cost of reduced energy savings of 31.5% for the instruction cache.

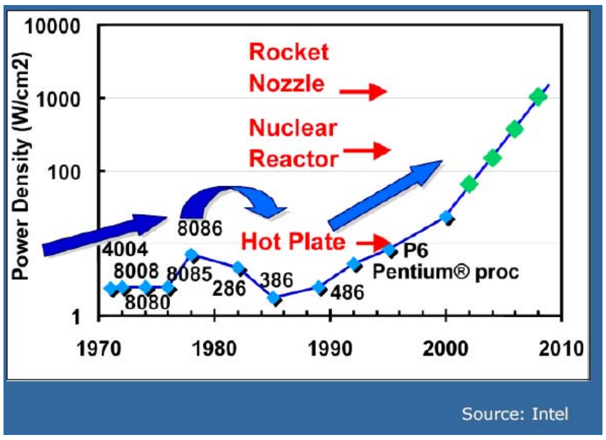

Figure 1. Extracted from Ref [13]

2. REQUIREMENTS OF PREDICTION TECHNIQUES FOR FILTER CACHE

The filter cache, a small auxiliary cache structure between the instruction cache and the execution core reduces power by trading performance. This cache hierarchy is well known as a method for reduction of instruction cache power. However, the performance degradation can be limited to acceptable levels by having a predictor, which predicts whether the next instruction to be fetched would be found (hit) in filter cache or not (miss).

Incorporation of the predictor leads to three major operating scenarios:

a. If the hit or miss prediction is correct then performance is maintained and power consumption is possible.
b. In the case of an incorrect hit-prediction, performance is sacrificed as the data is now fetched from L1 cache or a cache memory further away in the hierarchy.
c. In the case of an incorrect miss-prediction, where the data is actually in the filter cache but is predicted not to be, possible savings in power and, increased performance is sacrificed.

Thus it is important to have an accurate predictor, which is not pessimistic but realistic in its predictions. Furthermore, the power saving also depends upon the accuracy of the predictor and the assumptions made in the prediction algorithms.

A filter cache stores multiple instruction lines and it utilizes spatial and temporal locality. However, since it occupies the top-most level in the cache hierarchy, it is first accessed for every instruction fetch. Consequently, the small size of the filter cache results in higher miss rate, and thus performance is degraded.

3. NFPT VS. (PROPOSED) PATTERN BASED PREDICTION

In this section we briefly elaborate the popularly used NFPT predictor and the proposed Pattern Based prediction scheme.

3.1 NFPT Based Predictor

NFPT based prediction, proposed by Weiyu Tang et al, has been shown to achieve high prediction accuracy, thereby improving the performance with respect to hierarchical cache memory organization. However, the NFPT based predictor works on the assumption that the filter cache being small would be able to contain only small loops. Not surprisingly the NFPT based predictor is only able to capture temporal locality within small loops to predict whether the next instruction is expected to be in the filter cache. To predict, it also utilizes the fact that the same control path might be taken repeatedly.

In the event of a cache line change, if the tag of the current fetch address is equal to the tag of the predicted next fetch address based on the previous control path, as tabulated in the NFPT, then a hit in the filter cache is predicted. Otherwise the main cache is accessed. The assumption here is that, if the control path is in a small loop, the tags of the current fetch address and the predicted next fetch would be equal, and only if it is part of a small loop would it remain in the cache until the next iteration. Otherwise, it would be flushed out. Though this assumption holds true for small cache sizes, where only single small loops can be stored, it fails to work when the cache size increases, when it could hold more than one small or large loop. In such a case, the mechanism tends to wrongly predict a miss in the filter cache more often, thus accessing the main cache and eventually losing the advantage of the filter cache.

3.2 Pattern Based Predictor

Motivation for using pattern based predictor comes from the fact that though NFPT based predictor gives high prediction accuracy, its frequent inaccurate miss predictions result in expensive main cache access, though the instruction could've been located in the filter cache. This increases the instruction access power, and more importantly reduces performance in terms of average memory access cycle time.

Two level adaptive branch predictors have been researched quite extensively, and various kinds of two level pattern based predictors are being proposed [9] [10] [11]. Numerous studies have shown which predictors and configurations best predict the branches, in a given set of benchmarks. Some studies have also investigated effects, such as pattern history table interference, that can be detrimental to the performance of these predictors.

Since in our setup, we are only interested in predicting whether the next fetch source is the filter cache or not, we only require the pattern of the history of cache lines being present or otherwise in the filter cache. Hence, even a small shift register and pattern history table should be sufficient to reduce the interference.

Figure 2 shows the working of pattern-based predictor. It is based on the global two level adaptive branch predictor known as Gag [10]. Drawing from the insight provided by branch prediction researches, here we assume that next instruction fetch address will be current fetch address plus “4”. We access the predictor only when cache line changes are encountered. This is because we assume that most branches are not taken. Hence, we predict a hit in the filter cache if the cache line being accessed is unchanged. In the event of a change, we access the pattern history table corresponding to the shift register value, and the prediction of a hit or a miss in the filter cache is made depending on whether the pattern history table value reads above the pre-set threshold or below. For a two-bit saturating counter pattern history table, a threshold value of 2 is used. When the outcome of the prediction is known, the shift register and pattern history table values are accordingly updated, as in the normal branch predictor scheme.

Based on the above analysis the proposed Pattern based predictor is preferred over NFPT as an efficient alternative as it addresses the performance concern at the same time reducing the power losses. It is basically an alternative prediction scheme based on two-level adaptive correlated branch predictor, which we call as Pattern Based Predictor. This scheme relies on the reoccurrence of same patterns in the sequence of hits and misses in the program flow or instruction stream over the past *n*-cycles. The working of Pattern Based Predictor is explained in the next section. Following which it is shown that the Pattern Based predictor produces the best performance with respect to power and performance compared to the existing prediction techniques for most cases.

4. EXPERIMENTS

Simple Scalar [6] tool set with ARM ISA was used for these experiments. NFPT and pattern based predictors are implemented with the simple scalar tool set. To measure the relative access time and per access energy CACTI version 3.0 [12] was used with 0.18um technology. We simulated NFTP with full next fetch address for performance studies and 4-bit next fetch address as specified in the literature [2]

As shown in the published results the prediction accuracy will be of around 97% [2] of the figures given in this paper for NFPT implemented with 4-bit tags as all these results are with full next fetch address. This was done to compare the Pattern based predictor with best NFPT based configuration.

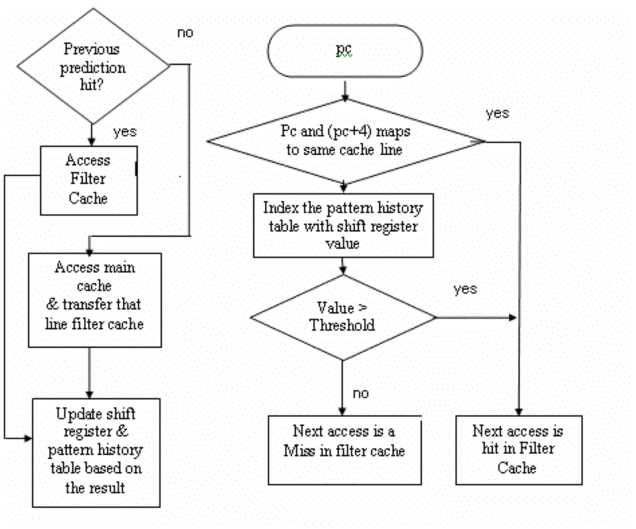
Previous prediction hit?
no
yes
Access Filter Cache
Access main cache & transfer that line filter cache
Update shift register & pattern history table based on the result
pc
Pc and (pc+4) maps to same cache line
yes
Index the pattern history table with shift register value
Value > Threshold
yes
no
Next access is a Miss in filter cache
Next access is hit in Filter Cache

Figure 2. Pattern Prediction Algorithm

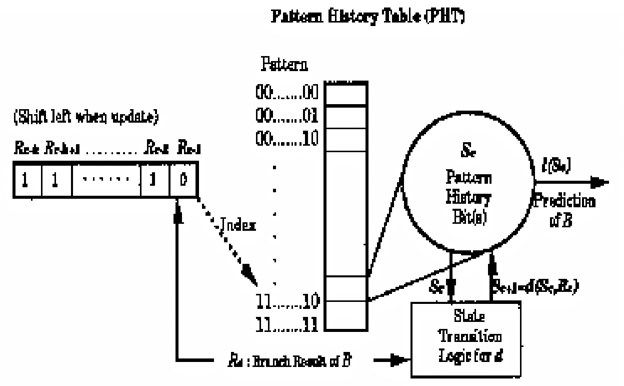

Figure 3. Working of two-level adaptive predictor

These experiments were based on 5-bit shift register and 32-entry (2^5) pattern history LUT with pattern based predictor as it more than sufficient to contain all the patterns required to predict the hit/miss pattern.

Mediabench [7] and Mibench [8] benchmark programs were selected as they characterize embedded applications working sets for high performance embedded processors. All the applications are compiled with gcc 2.93 with -03 optimization level and were run to completion.

Performance was measured as the prediction accuracy, i.e. the percentage of predictions that resulted in correct predictions. The other important comparison metric was the percentage cache access energy reduction, i.e. the percentage cache energy reduction on cache hierarchy for executing the particular program compared to the NFPT based approach. For all the experiments an 8KByte L1 cache with 32Byte line size and 32 way associativity was used.

4.1 Results

In this section we compare the prediction accuracy and percentage energy reduction achieved due to pattern-based predictor compared to the NFPT based predictor for 256, 512, 1024 Byte filter cache sizes.

Tables 1, 2, 3, and 4 shows the prediction accuracy and relative access energy of filter cache sizes of 256 Bytes, and 512 Bytes.

With 256 Bytes filter cache (FC) for the entire benchmarks pattern based predictor's prediction accuracy is better than NFPT based prediction accuracy and Pattern based predictor achieves better access energy saving except for Patricia and Susan bench marks with full next fetch address bits as described in the previous section. Average energy reduction achieved with pattern based predictor for the entire benchmark is 8.90 % with 256 Byte filter cache.

For 512 Byte filter cache, pattern predictor achieves better prediction accuracy except for Dijkstra, Ispell Patricia, and Susan. Pattern based predictor achieves better access energy saving except for Patricia, Dijkstra and Susan bench marks for these two filter cache sizes. The average energy savings due to pattern predictor for this benchmark is 4.8 %.

Benchmark	NFPT	Pattern based
ADPCM Encoder	98.01	99.97
ADPCM Decoder	97.51	99.98
Dijkstra	90.40	92.67
Ispell	92.17	92.55
jpeg decoder	95.49	97.76
Jpeg encoder	92.55	95.89
Patricia	91.65	91.98
Qsort	88.36	91.27
Susan	99.24	99.27

Table 1: Prediction accuracy (%) for 256 B FC

Benchmark	% Energy Savings due to pattern predictor
ADPCM Encoder	21.62
ADPCM Decoder	25.82
Dijkstra	0.43
Ispell	0.42
jpeg decoder	7.67
Jpeg encoder	14.85
Patricia	-0.86
Qsort	10.24
Susan	-0.13

Table 2. Instruction access energy % change for pattern based predictor for 256 B FC

Benchmark	NFPT	Pattern based
ADPCM Encoder	98.0251	99.99
ADPCM Decoder	99.9752	99.98
Dijkstra	97.0689	95.13
Ispell	93.164	92.71
jpeg decoder	96.0977	97.53
Jpeg encoder	95.349	98.05
Patricia	92.0956	91.94
Qsort	90.9285	94.04
Susan	99.3276	99.26

Table 3. Prediction accuracy (512 B FC)

Benchmark	NFPT
ADPCM Encoder	20.33
ADPCM Decoder	0.01
Dijkstra	-12.81
Ispell	-1.07
jpeg decoder	7.86
Jpeg encoder	18.12
Patricia	-0.59
Qsort	11.56
Susan	-0.25

Table 4. Instruction access energy % change for pattern based predictor for 512 B FC

Percentage power reduction Due to pattern predictor	256 Filter cache	512 Filter cache	1024 Filter cache
Maximum	25.82	20.33	13.31
Average	8.90	4.8	3.9
Minimum	-0.86	-12.81	-1.11

Table 5. Max/Min/Avg Power Savings over diff. Benchmarks for various Filter cache sizes

From tables 1-5 it is quite clear that the average energy savings reduces as the filter cache size is increased.

It is also evident that the NFPT based predictor requires significantly more area to implement the prediction algorithm and this will also increase with increase in filter cache size where as with pattern based predictor it is constant for particular shift register size and pattern history table.

5. CONCLUSIONS

In this paper we propose a prediction algorithm, which notably improves energy efficiency as compared to NFPT based approach, to predict whether the next fetch will be hit or miss in filter cache. The scheme provides up to 25.82% energy efficiency as compared to the NPFT based scheme. The mechanism also achieves better prediction accuracy with respect to NFPT for most benchmarks over different filter cache sizes. It is also easily appreciated that the pattern based prediction can be easily implemented in hardware unlike NFPT based prediction as it only requires a shift register and a LUT. Finally, our investigation firmly concludes that the benefit of the filter cache is at its best when the size of the same is minimal (Table 5).

References:

[1] Kin, J.; Munish Gupta; Mangione-Smith, W.H. "*The Filter Cache: An Energy Efficient Memory Structure*" Microarchitecture, 1997. Proceedings. Thirtieth Annual IEEE/ACM International Symposium on , 1997

[2] Weiyu Tang; Gupta, R.; Nicolau, A. "*Design of a predictive filter cache for energy savings in high performance processor architectures*" Computer Design, 2001. ICCD 2001. Proceedings. 2001 International Conference on, 2001

[3] Anderson, T.; Agarwala, S. "*Effective hardware-based two-way loop cache for high performance low power processors*" Computer Design, 2000. Proceedings. 2000 International Conference on, 2000

[4] Bellas, N.; Hajj, I.; Polychronopoulos, C.; Stamoulis, G. "*Energy and performance improvements in microprocessor design using a loop cache*" Computer Design, 1999. (ICCD '99) International Conference on, 1999

[5] Lea Hwang Lee; Moyer, B.; Arends, J. "*Instruction fetch energy reduction using loop caches for embedded applications with small tight loops*" Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on, 1999

[6] www.simplescalar.com

[7] Chunho Lee, Miodrag Potkonjak and William H. Mangione-Smith; "*MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems*" in Proceedings of the 30th Annual International Symposium on Microarchitecture, pp. 330-335, Dec. 1997.

[8] D. Ernst, T. M. Austin, T. Mudge and R. B. Brown "*MiBench: A free, commercially representative embedded benchmark suite,*" In proeedings of the 4th annual IEEE International Workshop on Workload Characterization, pp. 3-14, Dec. 2001.

[9] J.E. Smith, "*A study of branch prediction strategies*" Proc 8th Ann. Int'l Symp. Computer Architecture (ISCA 81), IEEE CS Press, Los Alamitos, calif., 1981 pp.135-148

[10] T.Y.Yeh and Y.N.Patt, "*Alternative Implementation of Two-Level Adaptive Branch Prediction*" 19th Ann. Int'l Symp. Compute Architecture. (ISCA 92), IEEE CS Press, Los Alamitos, calif., 1992 pp.124-124

[11] S.T.Pan, K. So, and J.T.Rehman, "*Improving the Accuracy of dynamic branch Prediction Using Branch Correlation*" Proc. 5th Int'l Conf. Architecture Support for Programming Languages and Operating Systems (ASPLOS-V), IEEE CS Press, Los Alamitos, calif., 1992, pp. 76-84.

[12] P Shivakumar and N *Jouppi "An Integrated Cache Timing, Power and Area Model*", Tech. Report , Compaq Western Research Lab, Palo Alto, Calif., 2001/2

[13] Fred Pollack "*New Micro architecture Challenges in the Coming Generations of CMOS Process Technologies*" Micro32 Conference, Haifa Israel. Nov'99