LETTER

Data Filter Cache with Partial Tag Matching for Low Power Embedded Processor*

Ju Hee CHOI[†], **Jong Wook KWAK**[††a)], **Seong Tae JHANG**[†††], ***Nonmembers***, ***and*** **Chu Shik JHON**[†], ***Member***

SUMMARY Filter caches have been studied as an energy efficient solution. They achieve energy savings via selected access to L1 cache, but severely decrease system performance. Therefore, a filter cache system should adopt components that balance execution delay against energy savings. In this letter, we analyze the legacy filter cache system and propose Data Filter Cache with Partial Tag Cache (DFPC) as a new solution. The proposed DFPC scheme reduces energy consumption of L1 data cache and does not impair system performance at all. Simulation results show that DFPC provides the 46.36% energy savings without any performance loss.
key words: *filter cache, L0 cache, data cache, partial tag, partial address*

1. Introduction

Cache memory systems generally consume 12~45% of processor energy [1]. As handheld devices become widespread, embedded system designers have focused on reducing the energy consumption of the cache. The filter cache system was proposed as an approach to alleviate the energy problem [2]. A small additional cache memory, which is called filter cache or L0 cache, is newly placed between core and L1 cache. In this system, the core fetches instructions or data from only the filter cache instead of the L1 cache. Because filter cache accesses consume much less power than L1 cache accesses, the energy dissipated by the entire chip can be reduced. Recently, due to the increasing demand of extremely low-power systems, modern processors have employed the filter cache system [3].

The original filter cache caused more than 20% performance degradation, although 51% reduction in the Energy-Delay Product (EDP) is achieved [2]. Because the performance degradation may not be acceptable in modern high-performance embedded environments, researchers have studied ways of addressing this weakness [4]–[6]. These studies have focused on mitigating miss penalty of filter cache or increasing the hit rate of filter cache.

In order to reduce miss penalty of the filter cache, a predictive filter cache has been proposed [4]. In this structure, a predictor is added to the filter cache system and the predictor selects the access path between L1 cache and filter cache. In addition, new policies about cache placement have been introduced to increase hit rate of L1 cache or filter cache [5], [6]. According to the policies, a cache line is inserted into either L1 cache or filter cache on a cache miss. Cache lines are also transferred from L1 cache to filter cache or from filter cache to L1 cache by the policies.

To the authors' best knowledge, no previous work has examined the difference in access latency between filter cache and L1 cache. In all previous works, the access latencies were assumed to be the same or they were not shown clearly in their studies. However, considering the size gap of filter cache and L1 cache in the modern embedded computing environments, it is possible that the access latencies are different. Thus, the filter cache system with multiple-cycle-latency needs to be explored and an alternative filter cache model should be built based on this analysis.

In this letter, we propose a new filter cache model using partial tag matching for energy efficiency. This model employs another small cache, called a partial tag cache, which is accessed simultaneously with the filter cache and used for blocking unnecessary L1 cache accesses. By minimizing the number of accesses to the L1 cache, it leads to the reduction of energy consumption.

2. Analysis of Filter Cache System

Since filter cache system was concentrated on reducing energy consumption of the cache access, performance was somewhat sacrificed. When filter cache system was originally proposed, it was assumed that hit latencies of L1 cache and filter cache were one cycle. If a filter cache miss occurs, the system takes one more cycle to access the L1 cache and it causes performance degradation. Therefore, recent researches about filter cache have been focused on the speed up of execution time. They presumed that reducing execution delay was the most important issue in the filter cache system.

However, modern processor systems gradually employ multiple-cycle-latency L1 cache. Unlike the system with single-cycle-latency L1 cache, it is possible that the performance of the cache system is not lower than that of the baseline system which has no filter cache. Assuming L1 cache requires 2 cycles and filter cache does 1 cycle as a hit latency, thus three consecutive accesses to L1 cache without a cache miss take 6 cycles in the baseline architecture. In the filter cache system, however, 2 hits and even 1 miss of filter

Manuscript received September 27, 2013.
Manuscript revised December 21, 2013.
[†]The authors are with the Department of Computer Science and Engineering, Seoul National University, Korea.
[††]The author is with the Department of Computer Engineering, Yeungnam University, Korea.
[†††]The author is with the Department of Computer Science, The University of Suwon, Korea.
*This work was supported by the 2013 Yeungnam University Research Grant.
a) E-mail: kwak@ynu.ac.kr
DOI: 10.1587/transinf.E97.D.972

cache take only 5 cycles. Because of the difference between hit latency of two cache systems, the latter is faster, even though one filter cache miss occurs.

Therefore, if miss rate of filter cache is relatively low to compromise hit latency of L1 cache, it is possible to achieve the reduction of energy consumption of the cache access without any performance degradation. In addition, it can even provide the reduction of execution time when reasonable hit rate of filter cache is provided.

To confirm that data filter cache system itself does not make system performance worse, we simulate the execution delay and the energy consumption using the *gem5* simulator [7]. We make a comparison between IPCs of the baseline architecture and the filter cache architecture to estimate the performance of two systems. As shown in Fig. 1, the energy consumption using the data filter cache is reduced by 44.38% on average and overall performance is even improved by 0.2%.

We divided the total cache access energy into the L1 data cache access energy and the data filter cache access energy as depicted in Fig. 2. Although the access counts of data filter cache exceed the access counts of L1 data cache, the L1 data access energy still dominates the whole cache access energy, as 85.72% on average. It is expected that reducing the number of L1 data cache accesses leads to the decrease of the total energy value. We will hereafter refer to L1 data cache as L1 cache.

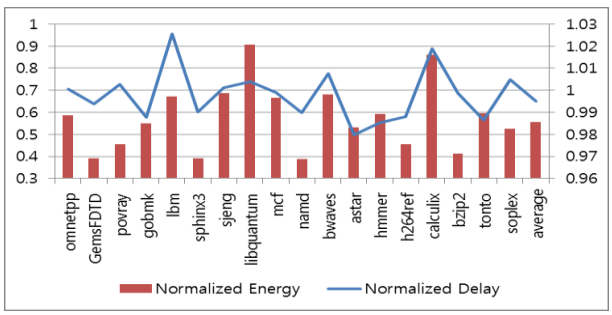

Fig. 1 Normalized energy and normalized delay for filter cache system, compared to the baseline cache system which has no filter cache; the left Y axis measures normalized energy, while the right Y axis illustrates normalized delay.

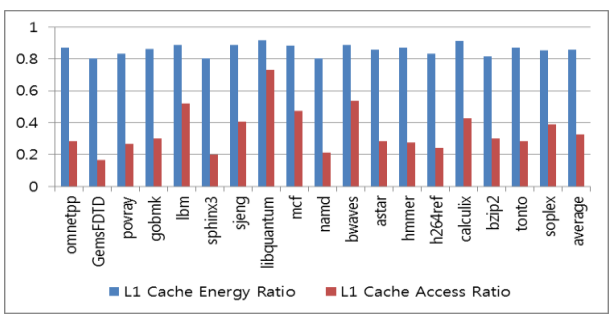

Fig. 2 L1 cache energy ratio and L1 cache access ratio.

$$\text{L1 cache energy ratio} = \frac{\text{L1 cache energy}}{\text{L1 cache energy+filter cache energy}}$$

$$\text{L1 cache access ratio} = \frac{\text{L1 cache access}}{\text{L1 cache access+filter cache access}}$$

3. Data Filter Cache System with Partial Tag Cache

In order to reduce L1 cache access energy, Data Filter Cache System with Partial Tag Cache (DFPC) is proposed. In DFPC, we employ a small device which contains partial tag information as illustrated in Fig. 3. The main role of the device is deactivating unnecessary ways of L1 cache during L1 cache access. In addition, we modify the tag array of L1 cache. Because the device has a part of tag address, each tag array entry has the rest of tag bits to save area and access energy.

One method to minimize the energy consumption of L1 cache access is reducing the number of way accesses of L1 cache. If it is revealed that the required data is not located in each way of L1 cache before L1 cache access, accessing such ways can be skipped. Therefore, in order to filter access to unnecessary ways, we employ a small cache which is called "Partial Tag Cache" as referred to PTC. It contains partial tag information of each way of L1 cache.

The PTC is accessed during filter cache access simultaneously. Thus, we can complete the comparison of the output of the PTC and partial tag of current data address before L1 cache access. This means that additional cycles are not needed for the PTC access. Therefore, we decide in which ways should be activated during L1 cache access.

When a filter cache miss occurs, our system predicts whether each way of L1 cache has the data or not by using this information. For example, assumed that L1 cache is a 4-

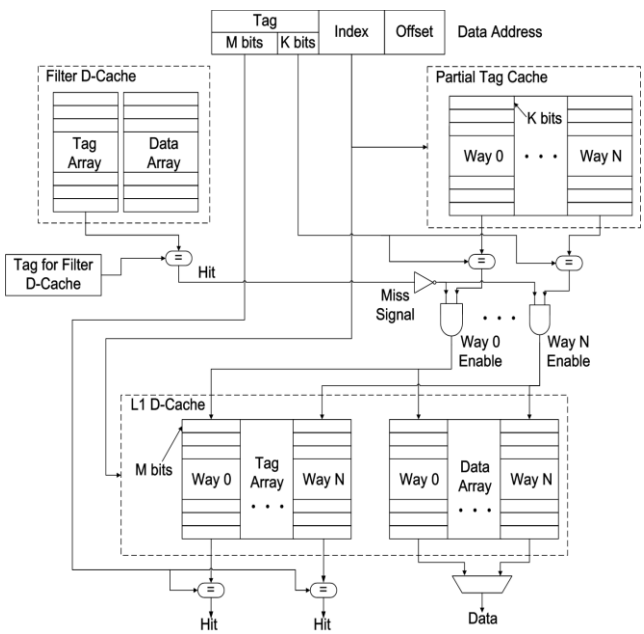

Fig. 3 Overall structure of Data Filter Cache with Partial Tag Cache. Each way of L1 data cache is only activated when Partial Tag of the way in the Partial Tag Cache is identical to the Partial Tag (K bits) of data address and a data filter cache miss occurs. Each tag entry of L1 data cache has the rest of tag (M bits) instead of full tag bits. The PTC is accessed during filter cache access simultaneously.

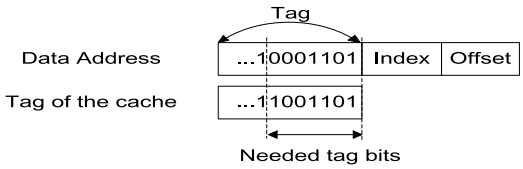

Fig. 4 Needed tag bits; To check whether these tags are different, at least 7 bits are needed, for 6 bits of two tags are identical.

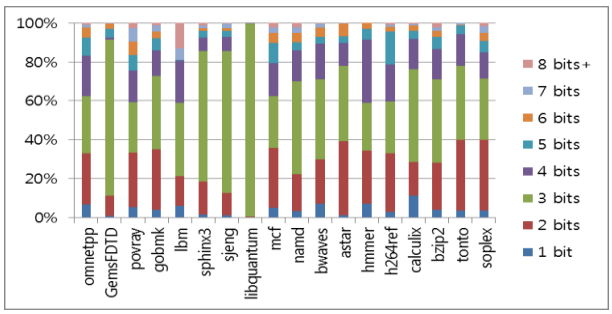

Fig. 5 The distribution of needed tag bits.

way set-associativity cache, if partial tags of way-0 and way-1 matches the partial address of the data address, we do not need to activate way-2 and way-3 of L1 cache. If there is no way to be activated, we can skip L1 cache access. Avoiding L1 cache access leads to the reduction of L1 access power as well as L2 cache hit latency.

We stores partial tag of L1 cache, instead of total tag information. Because it is known that tag matching by using a few bits, instead of whole tag information, is mostly enough, but a small cache with partial tag matching, which is less than 32KB, cannot avoid prolonging a critical path of the L1 cache access. However, unlike the previous work [8], no logic is inserted into the critical path of the L1 cache access in the DPFC. Thus, it does not induce any performance loss.

Then, an issue that should be considered is how many tag bits are required for partial tag matching. The needed tag bits are defined as the number of bits which are identical from LSB location as described in Fig. 4. If a small number of partial tag bits are used during tag matching, false hit in L1 cache may occur; partial tags are same, but full tags are different. A false hit causes unnecessary L1 cache access. On the contrary, if the number of tag bits is large, we can reduce the number of false hits but there is a probability that unnecessary tag bits will be additionally used.

Figure 5 shows that 3-bit tag matching covers 60% of whole tag matching, while 6 bit-tag matching covers 90%. The PTC access power is much less than L1 access power, therefore we utilize 6-bit tag matching scheme.

4. Performance Evaluation

4.1 Simulation Environments

In this letter, we simulate our approach with SPEC2006 benchmark suite [9]. The *gem5* simulator is used to evaluate the performance and the energy of our proposal [7]. The *gem5* is a full system simulator including a detailed model of an out-of-order SMT-capable CPU. In addition, CACTI 6.5 cache model is applied, with the technology of 0.32 nm, to estimate the dynamic energy of our proposal [10]. The overall simulation parameters are shown in Table 1.

Table 1 Parameters of the simulated architecture.

Parameter	Value
L0 Data Cache	1KB, direct-mapped, 64B line, 1 cycle hit
L1 Inst / Data Cache	32KB, 4-way, 64B line, 2 cycle hit
L2 Unified Cache	512KB, 16-way, 20 cycle hit
Memory	64bit bus width , 4 read/write ports
Function Units	6 IALU, 2 IMULT, 4 FPALU, 2 FPMULT

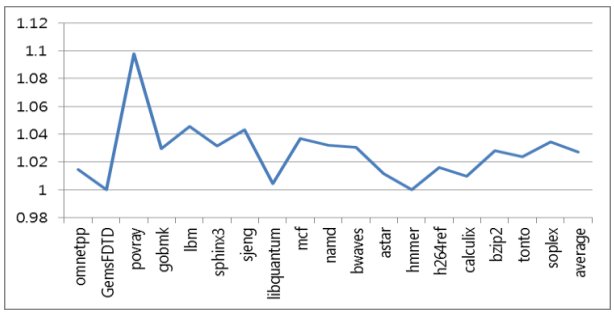

Fig. 6 The average number of way access per L1 access.

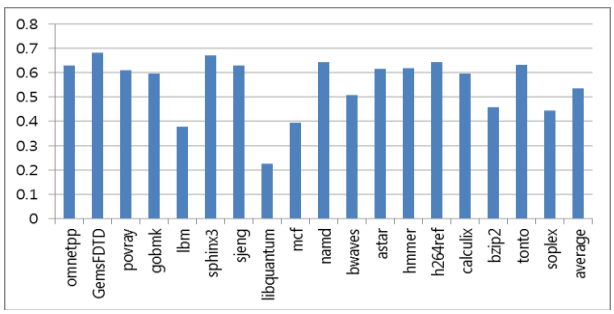

Fig. 7 Normalized energy for DFPC, compared to the original filter cache system.

4.2 Simulation Results

Figure 6 presents how many ways are activated for every L1 cache access. In simulation results, 1.027 ways on average are activated per memory reference in 4-way L1 cache. Therefore, almost 1-way access energy is consumed for every cache access, instead of 4-way access energy by using PTC system.

Figure 7 represents the simulation result of normalized energy for DFPC, compared to the filter cache system. The energy in the result means the sum of L1 cache access energy and filter cache access energy. Our system achieves 77.43% energy savings at maximum and 46.36% energy savings on average.

Next, in order to examine main factors for this reduction of DFPC energy consumption, we investigate L1 miss

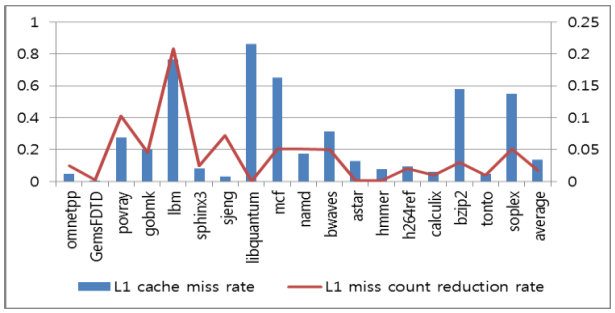

Fig. 8 L1 cache miss rate and L1 miss count reduction rate; the left Y axis measures L1 miss rate, while the right Y axis illustrates L1 miss count reduction ratio.

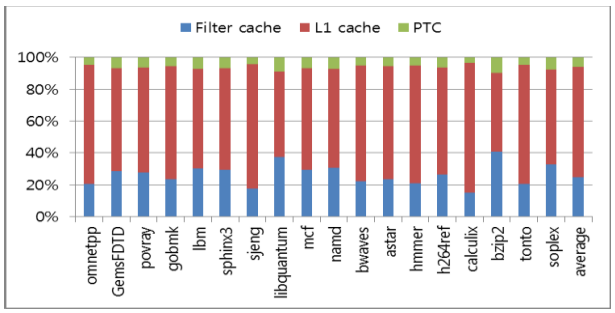

Fig. 9 The energy breakdown of the DFPC.

rate of the legacy filter cache system and L1 miss count reduction rate, as shown in Fig. 8. The L1 miss count reduction rate means that how many miss counts are reduced in DPFC, compared to legacy filter cache system. Overall, the L1 miss rate is proportional to the L1 energy reduction rate, thus the L1 miss rate can be considered as a dominant factor of the energy reduction of DFPC. In addition, the reduction rate of L1 miss count is another factor to influence the energy consumption. As depicted in Fig. 8, some benchmarks, such as *libquantm* and *lbm*, have the similar L1 miss rates, but their energy reduction rates are quite different. Therefore, we conclude that L1 miss rate and L1 miss count reduction rate are important factors to impact on the energy consumption.

Finally, we estimated the overhead of the PTC, as shown in Fig. 9. The PTC consumed 5.84% of the total energy consumption of DFPC. Although the PTC is accessed every data filter cache access, the overhead of the energy consumption is almost trivial. Therefore, we conclude that the reduction in total energy consumption of the DFPC system dominates the extra energy consumed in the PTC.

5. Conclusion

In this letter, Data Filter Cache with Partial Tag Cache scheme was proposed for energy efficient but high performance embedded system. To get new insights on the filter cache mechanism, we investigated the legacy filter cache system. The analysis showed that the filter cache system did not degrade performance in multi-cycle-latency L1 cache system. In addition, the L1 data cache is found to be responsible for a significant part of the power disspation in the filter cache system. As a result of this analysis, we focused on reducing the number of accesses to the L1 data cache. Our model exploited a Partial Tag Cache which stores partial tag information. The partial tag cache is accessed simultaneously with the data filter cache and used for blocking unnecessary L1 cache accesses. Thus, minimizing the number of accesses to L1 cache led to improvements in the energy consumption without performance loss. The simulation result showed that the reduction of average energy consumption in DFPC is 46.36% and 77.43% at maximum.

References

[1] A. Sodahi, "Race to exascale: Opportunities and challenges," MICRO 2011, Keynote talk.
[2] J. Kin, M. Gupta, and W.H. Mangione-Simith, "The filter cache: An energy efficient memory structure," Proc. 30th MICRO, pp.184–193, 1997.
[3] B. Klug and A.L. Shimpi, "Qualcomm's new snapdragon S4: MSM8960 & krait architecture explored," http://www.anandtech.com/show/4940/qualcomm-new-snapdragon-s4-msm8960-krait-architecture/2, Oct. 2011.
[4] W. Tang, A. Kejariwal, A. Veidenbaum, and A. Nicolau, "A predictive decode filter cache for reducing power consumption in embedded processors," ACM Trans. Design Automation of Electronic Systems, vol.12, no.2, pp.1–14, April 2007.
[5] N. Duong, T. Kim, D. Zaho, and A. Veidenbaum, "Revisiting level-0 caches in embedded processors," Proc. Int. Conference on Compilers, Architectures and Synthesis for Embedded Systems, pp.171–180, 2012.
[6] J. Tao, D. Hillenbrand, L. Wang, and H. Marten, "Studying filter cache bypassing on embedded systems," IEEE 10th Int. Conf. on Computer and Information Technology, pp.1679–1686, 2010.
[7] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill, and D.A. Wood, "The gem5 simulator," ACM SIGARCH Computer Architecture News, 2011.
[8] R. Min, Z. Xu, Y. Hu, and W.B. Jone, "Partial tag comparison: A new technology for power-efficient set-associative cache designs," 17th Int. Conference on VLSI Design, pp.183–188, Jan. 2004.
[9] J.L. Henning, SPEC CPU2006 Benchmark Descriptions, http://www.spec.org/cpu2006/publications/CPU2006benchmarks.pdf, 2006.
[10] N. Muralimanohar, R. Balasubramonian, and N.P. Jouppi, "CACTI 6.0: A tool to model large caches," HP Laboratories, Tech. Rep. HPL-2009-85, 2009.