# Counter-based Victim Cache Hit Rate Optimization

Li-Yen Chang s1003308@mail.yzu.edu.tw
Chen-Hua Suo s1016042@mail.yzu.edu.tw
Yi-Yu Liu yyliu@saturn.yzu.edu.tw

Department of Computer Science and Engineering, Yuan Ze University
Taoyuan, Taiwan 320, R.O.C.

**Abstract— Victim cache is proposed to alleviate cache miss penalty due to conflict misses. However, the low hit rate of victim cache, due to its small-size nature, indicates inefficiency of accessing the victim cache. In this paper, we integrate a small counter into L1 cache entry to filter out unnecessary victim cache accesses. Experimental results demonstrate that our technique substantially improves the victim cache hit rate.**

## I. Introduction

Victim cache (VC) is a small fully-associative cache which stores evicted cache blocks. If a cache block is accessed in a VC, the cache block is then moved back to the original cache. Therefore, VC provides a second chance to an evicted cache block. However, the hit rate of VC is quite low owing to its small size. In this work, we propose a counter-based technique to improve VC hit rate. The rest of this paper is organized as follows. Section II briefly gives the background and motivation of this work. The counter-based cache architecture is proposed Section III. Section IV lists the preliminary experimental results. The summary and ongoing work is given in Section V.

## II. Background and Motivation

As the increasing performance gap between processor and memory, multi-level cache hierarchy exploits spatial and temporal localities to provide an illusion of a large and fast memory system. In order to reduce conflict misses incurred in a direct-mapped cache, set-associative cache allows multiple data blocks coexisted in the same cache entry. However, it is impractical to unlimitedly increase the associativity (e.g., fully-associative) of a cache entry owing to the extra tag comparison logic and cache block multiplexing datapath overhead [1].

Conflict miss is one of the important performance-loss factors in memory hierarchy. Besides of set-associative cache architecture, several state-of-the-art multi-lateral cache architectures are proposed [2]- [5]. Miss cache is a small fully-associative cache as a temporal storage for new cache block. The cache block can be moved (upgrade) to L1 cache only when there is another cache access on that block. The behavior of VC is similar to that of a miss cache. The only difference is that VC stores evicted cache block (downgrade) from L1 cache [2]. In addition to the simple upgrade and downgrade approaches, Tam et al. propose a hardware-based scheme to keep track of active conflicts between cache blocks and to prevent conflicts by adjusting new cache block placement policy [3].

In order to exploit data access concurrency in L1 cache and VC, Bahar et al. propose a parallel VC architecture to improve the overall system performance [4]. However, a VC miss definitely occurs when there is a hit in L1 cache. This leads to an unnecessary VC access and results in wasting the dynamic power consumption. Furthermore, if there is no evicted cache block stored in a VC, there is no need to access the VC [5]. This motivates us to design a more accurate scheme to prevent unnecessary VC accesses.

## III. Counter-based Victim Cache

Since the parallel VC results in tremendous unnecessary VC accesses, our counter-based VC is designed based on sequential VC. There is an additional counter in each cache entry. The value in a counter indicates the number of evicted cache blocks (with same cache index) stored in current VC. If the value is 0, we don't need to access VC on a cache miss; otherwise we need to check if the missed cache block stored in VC. In order to keep the values up-to-date, we need to consider the combinations of three events: L1 evict, VC hit, and VC evict. (1) On L1 evict and VC miss, the counter is increment by 1. (2) On VC evict, the counter is decrement by 1. (3) On L1 evict and VC hit, the counter is unchanged. Fig. 1 briefly illustrates our proposed counter-based VC.

## IV. Preliminary Experimental Results

The preliminary experiment is conducted using a full system simulator - Simics [6]. Benchmark programs are selected from SPEC CPU 2006, Media Bench and Parsec benchmark suites. The experimental environment is listed in Table I. Fig. 2 summarizes the hit rates of various VC architectures. The geometric mean of counter-based VC hit rate is 85.9%, while the hit rate of sequential and parallel VCs are 54.7% and 1.6%, respectively.

Fig. 3 compares the access counts of victim caches. According to our results, counter-based VC is capable of reducing 36% unnecessary VC accesses as compared to sequential VC.

Finally, Fig. 4 compares the overall performance in terms of instructions per cycle (IPC). As illustrated in Fig. 4, the overall performance of counter-based VC is similar to those of parallel and sequential VCs within 0.5%.

## V. Summary and Ongoing Work

We have proposed a counter-based victim cache optimization technique to prevent unnecessary victim cache accesses. According to preliminary experimental results, our approach can substantially improve victim cache efficiency in terms of access count and hit rate. The overall performance in terms of IPC is slightly reduced within 0.5% as compared to the parallel victim cache architecture. We are working on energy consumption comparison and counter hardware cost reduction.

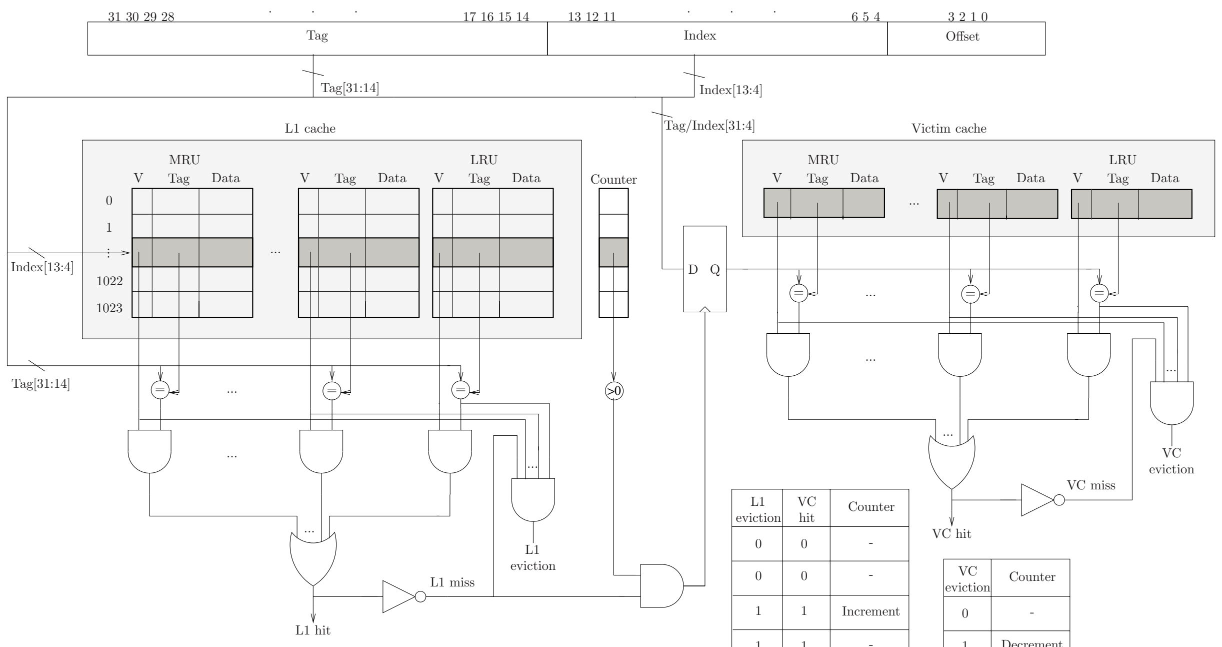

Fig. 1. Counter-based Victim Cache

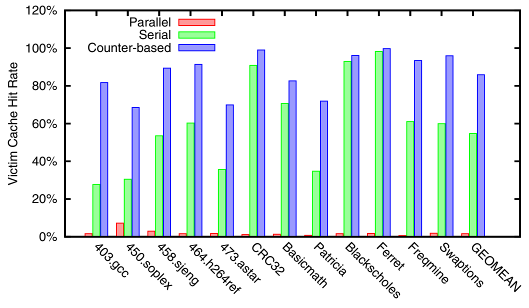

Fig. 2. Comparisons on Victim Cache Hit Rates

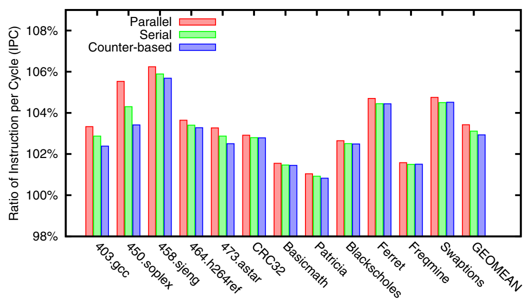

Fig. 4. Comparisons on Instruction per Cycles (IPC)

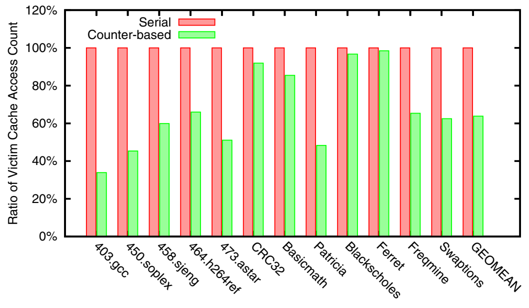

Fig. 3. Comparisons on Victim Cache Access Count

TABLE I
EXPERIMENTAL ENVIRONMENT

| **Target Machine** |
|---|
| Simulator: Simics-3.0.31<br>1GHz Tango(x86) CPU<br>In-order Execution mode |
| VC: 1KB, fully-assoc., block size 64B<br>L1: 8KB, direct, block size 64B<br>L2: 256KB, 4-way, block size 64B<br>Memory: 256MB |

## REFERENCES

[1] B. Jacob, S. Ng, D. Wang, "Memory Systems: Cache, Dram, Disk", Morgan Kaufmann, 2007.

[2] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers", *in Proceedings of International Symposium on Computer Architecture*, pp. 364-373, 1990.

[3] E. S. Tam, S. A. Vlaovic, G. S. Tyson, E. S. Davidson, "Allocation by conflict: A simple, effective multilateral cache management scheme", *in Proceedings of International Conference on Computer Design*, pp.133-140, 2001.

[4] I. Bahar, B. Calder, D. Grunwald, "A comparison of software code reordering and victim buffers", *in ACM SIGARCH Computer Architecture News*, Vol. 27, I. 1, 1999.

[5] G. Memik, G. Reinman, W. H. Mangione-Smith, "Reducing energy and delay using efficient victim caches", *in Proceedings of International Symposium on Low Power Electronics and Design*, pp. 262-265, 2003.

[6] Wind River Simics, http://www.windriver.com/simics/.