Novel Efficient Synonym Handling Mechanism for Virtual-real Cache Hierarchy

Varun Venkitaraman* Ashok Sathyan* Shrihari P. Deshmukh* Virendra Singh*
* Indian Institute of Technology Bombay

***Abstract*—Optimizing L1 caches for latency is critical to improving the system's performance. Generally, virtually indexed physically tagged (VIPT) caches are preferred L1 cache configuration because we can perform address translation and set indexing parallely, resulting in reduced L1 cache access latency. However, an address translation is essential for every L1 cache access. Address translation significantly contribute to the system's total power consumption. To reduce power consumed due to address translation, virtually indexed virtually tagged (VIVT) caches appears to be an attractive alternative. However, VIVT caches are plagued with the issue of synonyms. Prior works introduce new hardware structures in the cache hierarchy to detect and resolve synonyms. Rather than adding extra hardware structure to the cache hierarchy, we propose a new cache hierarchy design that modifies the last-level cache's tag array to detect and resolve synonyms. Our proposed scheme enhances system's performance by 22% on average and also reduces the dynamic energy consumption of the cache hierarchy by as much as 89%.**

***Index Terms*—Virtual caches, Energy-efficiency, Synonyms, Cache memories**

I. Introduction

Modern processors optimize L1 caches to enhance performance indifferent to energy consumption. Consequently, the local cache hierarchy significantly contributes to the processor's total core power consumption. Intel's study [1] states that the caches consume about 12% - 45% of the total core power. Typically, designers choose VIPT L1 caches. However, in case of VIPT L1 caches, a Translation lookaside buffer (TLB) access is necessary for every L1 cache access. According to Intel [1], the TLBs consume about 3% - 13% of the total core power. Decreasing the number of TLB lookups will result in reduced energy consumption due to TLB accesses. To accomplish this, a viable approach would be to use VIVT L1 caches instead. Regardless of the benefits, VIVT caches suffer from the issue of synonyms which negatively impacts the performance. Prior studies propose various techniques to resolve synonyms for facilitating the implementation of VIVT L1 caches [2]–[4]. In general, previous proposals use additional hardware structures to detect and resolve synonyms. However, introducing additional hardware structures to handle synonyms will increase the critical path length for L2 cache-sensitive applications leading to system performance degradation. Furthermore, Yoon et al. [3] demonstrated that for small cache sizes, typical of L1 caches, the number of pages with active synonyms is relatively low. Only a small number of L1 cache accesses come from pages with active synonyms. Leveraging these observations, we propose an energy-efficient speculation-free synonym handler in this work that alleviates latency overheads for synonym handling. Instead of adding new structures, we modify the existing L2 cache tag array by augmenting the location information of all the cache blocks present in the cache hierarchy to detect and resolve synonyms.

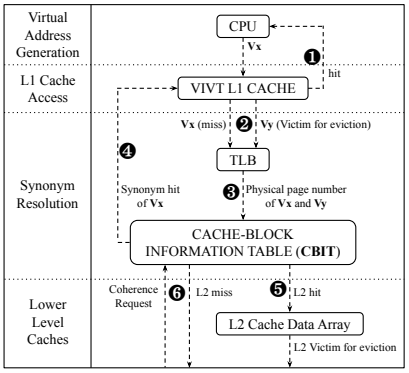

Fig. 1. Proposed Virtual-Real Cache Hierarchy

II. Proposed Scheme

Fig. 1 depicts the proposed two-level cache hierarchy where V_x and V_y are synonyms. The core accesses the VIVT $L1$ cache using **v**irtual **a**ddress (VA) V_x. On an $L1$ cache hit, the core receives the data for VA V_x, without the need for address translation (Fig. 1❶). When a cache miss occurs, it can either be due to a synonym or genuine $L1$ cache miss. To identify the type of cache miss, we propose a synonym handler, **c**ache-**b**lock **i**nformation **t**able (CBIT). Here, CBIT is the modified L2 tag array which is used to detect and resolve synonyms. On an L1 cache miss, the VA is translated to **p**hysical **a**ddress (PA) using the TLB (Fig. 1❷). CBIT is accessed using this PA (Fig. 1❸). The CBIT, indexed using the PA, tracks all the cache-blocks present in both the $L1$ and the $L2$ caches. If there is a CBIT miss, then the memory access is not a synonym miss. In the case of a CBIT hit, there are two possible scenarios. First, if the CBIT entry points to the $L1$ cache, then the $L1$ cache miss is due to synonym access (Fig. 1❹). The core reads the data from the $L1$ cache by using the location information given by CBIT. There is no need for tag comparisons for $L1$ synonym re-access (Fig. 1❹). Second, if the CBIT entry points to the $L2$ cache, then it is a genuine $L1$ cache miss (Fig. 1❺). Here, the core reads the required data from the $L2$ cache data array by using the location information given by CBIT. In case of a CBIT miss, the demanded cache-block is not present in the cache hierarchy (Fig. 1❻) and is brought from the main memory. To compact CBIT design, we analyzed the redundancy present in the L2 cache tags. For SPEC 2006 CPU benchmark suite, we observe that the maximum number

978-3-9819263-7-8/DATE23/© 2023 EDAA

TABLE I
BASELINE SYSTEM CONFIGURATION

Core	Single core, ROB = 192 entry, LQ = 32 entry, SQ = 32 entry, Fetch width = 4, Frequency = 3 GHz.
Caches	2 level hierarchy, Inclusive at all levels, Writeback, LRU replacement policy, 64B cache block size, Separate L1 instruction and data caches.
L1 - I/D Cache VIPT type	Size = 32 KB, Associativity = 8, Tag access latency = 1 cycle, Data access latency = 4 cycles.
L2 Cache (LLC) PIPT type	Size = 2 MB, Associativity = 16, Tag access latency = 7 cycles, Data access latency = 14 cycles.
RAM (main memory)	DDR3, Access latency = 45 ns, Queue delay modeled.

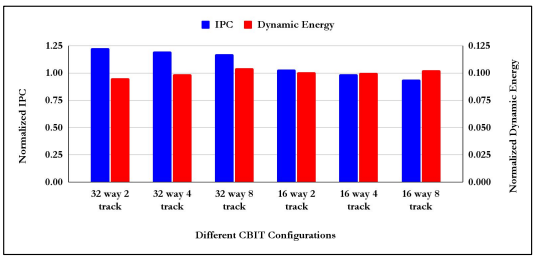

Fig. 2. System Performance & Dynamic Energy Analysis

of different tags present in L2 cache tag array range from 100 to 3700. Using this observation *we infer that most of the cache blocks accessed during the execution phase are from very few pages*. We leverage the observed redundancy in the L2 cache tags for compacting the number of entries in CBIT. We compact the CBIT size by tracking multiple consecutive cache-blocks of a page. Assuming that the $L2$ cache data array has m sets, the CBIT that tracks n consecutive cache-blocks of a page will contain m/n sets. Fig. 1 shows the overview of the proposed cache hierarchy where the CBIT tracks four consecutive cache-blocks of a page. The CBIT efficiently handles synonyms by reducing three main components: (1) the dynamic energy consumption due to tag comparisons of synonym re-access, (2) the latency of the $L1$ synonym re-access, and (3) the $L2$ access latency by directly providing the exact way number in $L1$ and $L2$ caches respectively. Though compacting CBIT provides latency and energy benefits, it has some side effects. Since the CBIT tracks multiple consecutive cache-blocks of a page using a single entry, a CBIT replacement can lead to multiple cache-block evictions in both the $L1$ and the $L2$ caches. Thus, we require a CBIT replacement policy to minimize the proposed scheme's impact on cache occupancy of both the $L1$ and the $L2$ caches.

III. EVALUATION

The latency and energy of different set-associative caches was modeled using Cacti 6.5 in 32nm technology node [5]–[7]. Table I shows the baseline configuration used for evaluating our proposed scheme. We use CPU SPEC2006 benchmark applications for evaluating the proposed architectural modifications [8]. We evaluate our proposed scheme using Gem5 x86 simulator [9] in full system emulation mode. The performance impact of the applications was studied by running the simulations from the checkpoints taken at twenty billion instructions. The applications were run for one billion instructions from the checkpoint for warm-up, and after that, we considered the performance over the next one billion instructions for evaluation.

TABLE II
AREA OF DIFFERENT CBIT CONFIGURATIONS

Configuration	Area (mm^2)	Configuration	Area (mm^2)
16-way 2-track CBIT	0.402	32-way 2-track CBIT	0.903
16-way 4-track CBIT	0.267	32-way 4-track CBIT	0.530
16-way 8-track CBIT	0.213	32-way 8-track CBIT	0.749

Fig. 2 shows that the 32-way-2-track CBIT configuration provides maximum performance benefits of 22% on an average across all CPU SPEC2006 benchmark workloads even while tracking two contiguous cache blocks in a page with a single CBIT entry. Due to space constraints, we provide nine applications from the benchmark suite that spans a wide range of behavior. The dynamic energy consumption of the cache hierarchy is shown in Fig. 2. We save around 89% cache hierarchy's dynamic energy by adopting our proposed scheme. The dynamic energy consumption of the cache hierarchy is comparatively higher for CBIT configuration with a large number of cache blocks tracked. The area of L2 data array is $5.54mm^2$, and that of the L2 tag array (for baseline) is $0.57mm^2$. In the proposed scheme, the L2 data array remains the same as that in the baseline, and the CBIT replaces the L2 tag array. Table II lists the area of CBIT for various configurations. For the state-of-the-art comparison, we model VC-DSR [3]. *We also observed that the proposed technique with 32-way-4-track CBIT configuration offers performance boost of 21% over VC-DSR on an average over all the applications in the benchmark suite*.

IV. CONCLUSION

The paper proposes an *efficient synonym handler, CBIT,* to facilitate VIVT L1 caches implementation, which is purely a hardware solution. The CBIT leverages the redundancy in tags present in the cache hierarchy. It detects and resolves all the active synonyms. *We have achieved up to* 89% *reduction in the cache hierarchy's dynamic energy consumption with a performance boost of* 22% *on average.*

REFERENCES

[1] A. Sembrant, E. Hagersten, and D. Black-Shaffer, "Tlc: A tag-less cache for reducing dynamic first level cache energy," in *MICRO*, 2013.

[2] J. R. Goodman, "Coherency for multiprocessor virtual address caches," in *ACM SIGARCH Computer Architecture News*, 1987.

[3] H. Yoon and G. S. Sohi, "Revisiting virtual l1 caches: A practical design using dynamic synonym remapping," in *HPCA*, 2016.

[4] X. Qiu and M. Dubois, "The synonym lookaside buffer: A solution to the synonym problem in virtual caches," *IEEE Transactions on Computers*, 2008.

[5] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

[6] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "Cacti-p: Architecture-level modeling for sram-based structures with advanced leakage reduction techniques," in *ICCD*, 2011.

[7] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP laboratories*, 2009.

[8] J. L. Henning, "Spec cpu2006 benchmark descriptions," *ACM*, vol. 34, no. 4, pp. 1–17, 2006.

[9] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," 2011.