A Space-Efficient Footprint DRAM Cache

Jongwon Kim, Yongjun Lee, Hakbeom Jang, and Jae W. Lee
Sungkyunkwan University
2066, Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do 16419, Korea
E-mail: {kimjongwon, yongjunlee, hakbeom, jaewlee}@skku.edu

Abstract: Recently, 3D die-stacked DRAM technologies have been adopted by many processor vendors as a solution to the memory wall problem. To effectively utilize this large in-package memory, there are proposals to use it as a cache. Page-based caches have drawn much attention to reduce the cost of tags by increasing the granularity of caching. However, page-based caches waste cache capacity by over-fetching blocks that are not actually used during the page's lifetime in the DRAM cache. In this paper, we introduce a novel footprint caching technique, which greatly improves the space efficiency of a page-based cache. The key idea is to overlay two sparse pages with many invalide blocks into a single physical page, thus maximizing page utilization. Our cache design improves the IPC by 17.9% and reduces cache miss by 5% over a state-of-the-art footprint cache.

Keywords—**Die-stacked DRAM , Memory footprint**

1. Introduction

Recently, to use effectively large in-package DRAM, page-based cache designs have been proposed to minimizing cache tag overhead. Unlike the conventional block-based caches (e.g., 64B block size), which require large tag storage and then incur long tag access latency, page-based caches increase the caching granularity to several KBs to reduce storage overhead and improve cache hit rate by better exploiting spatial locality.

Caching at page granularity alleviates the cache tag overhead, but the tag overhead still significant as DRAM cache size continues to grow. Lee et al. [2] propose Tagless DRAM Caches (TDCs), which align the granularity of caching with OS page size and employ the cache-map TLB (cTLB) to maintain virtual-to-cache address mappings directly. So a cache tag-checking operation is eliminated from the cache access path to yield lowest cache hit latency.

Page-based caches suffer an over-fetching problem. Caching at page granularity has benefit of high hit rate by allocating many data blocks at once, but sometimes unused blocks as well. These over-fetched blocks cause off-package bandwidth pollution and cache capacity wastes.

To address this problem, Jevdjic et al. [3] propose Footprint Cache to improve the bandwidth efficiency of page-based DRAM caches. Footprint Cache stores the memory footprints of evicted pages in a small Footprint History Table. Using this footprint Footprint Cache fetches only referenced blocks from the requested page. While Footprint

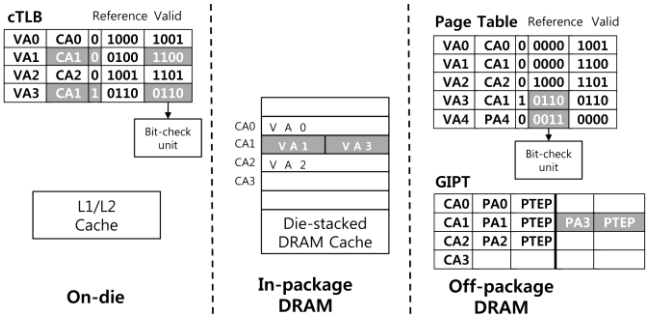


Figure 1. Space-Efficient DRAM cache architecture

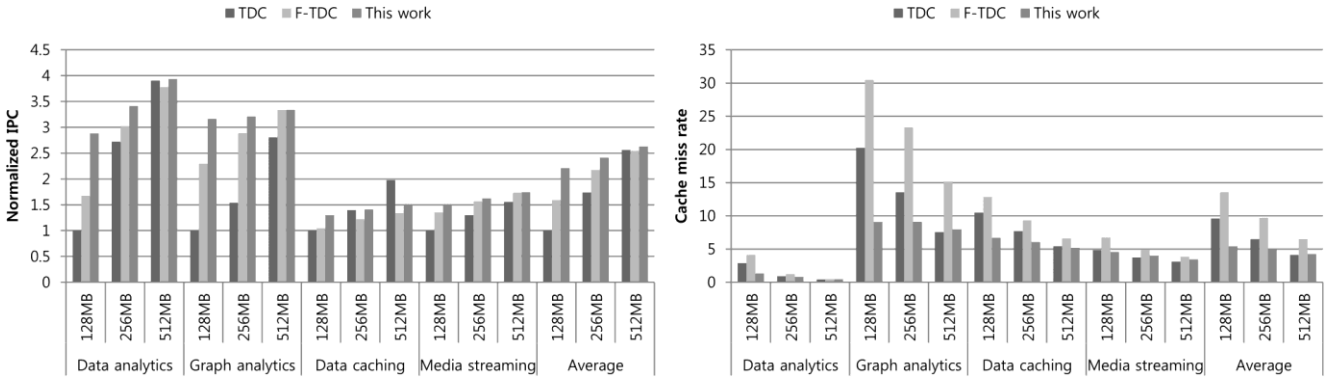
Cache mitigates bandwidth pollution, it still maintains expensive tags, thus scaling not as well as TDC. Jang et al. [1] propose Footprint-augmented Tagless DRAM Cache (F-TDC), which combines footprint caching with TDC to improve bandwidth efficiency. Since F-TDC store all memory footprints in the page table, F-TDC achieves higher hit rate with low hardware cost. However, their solution does not address the problem of wasting cache space. Gulur et al. [4] propose Bi-model Cache, which uses two different caching granularities: block (64B) or page (512B) granularities considering spatial locality and meta data of the page. By allocating blocks, cache space can be exploited more efficiently. However, the cache miss latency is still long as a page and a block accesses are serialized.

This paper introduces a space-efficient footprint cache, which allocates a half of the page when the footprint of the requested page indicates that valid blocks in the page can be contained in the half page. Thus, the proposed design improves both bandwidth and space efficiency.

2. Space-Efficient Footprint DRAM Cache

Figure 1 shows the overall structure of the proposed design. cTLB, page table and Global Inverted Page Table (GIPT) are modified from F-TDC.

Memory footprint: For footprint tracking, cTLB and page table store a reference bit vector and a valid bit vector. Each bit vector consist of 8 bits and 1 bit repersent 8 blocks. Reference bits indicate which blocks have been touched during previous page's lifetime to be used as fetching hints at next allocation. When a page is allocated in the DRAM cache, we check reference bits in page table to determine whether to allocate a half page or full page. Valid bits indicate which blocks are currently valid in the DRAM cache. When accesing the cache, we check valid bits in cTLB and the half-page bit.



(a) Normalized IPC (b) Cache miss rate

Figure 2. Nomalized IPC and Cache miss rate for TDC, F-TDC, and Space-Efficient Footprint Cache

Hardware Design: We use a bit-check unit, which determines whether the page is contained in a half-page. For example, if VA1 and VA3 of cTLB have valid block range of '2', VA1 and VA3 are allocated to a half-page. If the page is allocated to a half page, we obtain 1-bit information. The bit indicates the position of the half-page in the DRAM cache. Because two different half-pages are located in one physical page, GIPT also needs more storage. GIPT is extended to be a mapping structure that two page addresses to one cache address.

Operations: The basic operations of the proposed cache are similar to F-TDC. For every cache access, there is overhead to access the bit-check unit. In case of a page miss, the bit-check unit checks the reference bits of a requested page and then, if the page fits in a half page, the page is allocated to the half-page. In case of a block miss for half-page, the system checks the requested block. If the requested block cannot fit in half-page, generate page miss. And if the block exceeds half-page size, lookup GIPT and find other half page in same cache page. Next, migrate the other page and extend page that include requested block.

3. Evaluation

We evaluate performance of the proposed design using McSimA+ [5]. Table 2 summarizes architectural parameters. We use four benchmarks of CloudSuite [6]. Figure 2(a) shows the IPC of TDC, F-TDC and our proposal. Normalized IPC is better than F-TDC, but TDC has best performance for data caching and data analytics at 512MB. These workloads have small memory footprint to have a high reuse rate. However, as an application's size grows, F-TDC and our proposal get better performance than TDC like 128MB DRAM cache size. Our proposal achieves average normalized IPC improvement by 39.1%, 11.1%, and 3.3% over F-TDC with 128MB, 256MB, and 512MB DRAM caches. Figure 2(b) shows the cache miss rate of TDC, F-TDC, and our proposal. Total cache miss means page miss and block miss. TDC has no block miss case, so in the majority of cases TDC has lower cache miss than F-TDC. But our proposal can locate two pages in one cache page. Therefore, our proposal can reduce cache misses. Our proposal achieve 8.1%, 4.7%, and 2.2% lower cache miss rate than F-TDC at 128MB, 256MB, and 512MB.

Table 1. Architectural parameters

Component	Parameters
CPU	Out-of-order, 4cores, 3GHz
TLB	L1 32I/32D entries per core
	L2 512entries per core
Cache	L1 4-way 32KB I-cache/D-cache, 64B line, 2cycle
	L2 16-way, 2MB shared cache per core, 64B line, 6 cycle
In-package DRAM	1.6GHz (DDR 3.2GHz), 1channel, 2ranks, 16banks per rank, 128 bits bus width
Off-package DRAM	800MHz(DDR 1.6 GHz), 1channel, 2ranks, 64banks per rank, 64bits bus width

4. Conclusion

This paper introduces a space-efficient footprint DRAM cache, which supports half-page caching for pages with small footprints. Space-efficient footprint DRAM cache locates two pages on one page and can support more cache capacity. Therefore, our proposal obtains higher cache hit ratio and improves cache performance.

References

[1] H. Jang, Y. Lee, J. Kim, Y. Kim, J. Kim, J. Jeong, and J. W. Lee,"Efficient Footprint Caching for Tagless DRAM Caches," in Proc. 22nd International Symposium on High Performance Computer Architecture (HPCA), 2016.

[2] Y. Lee, J. Kim, H. Jang, H. Yang, J. Kim, J. Jeong, and J. W. Lee,"A Fully Associative, Tagless DRAM Cache," in Proc. 42nd Annual International Symposium on Computer Architecture (ISCA), 2015.

[3] D. Jevdjic, S. Volos, and B. Falsafi, "Die-Stacked DRAM Caches for Servers: Hit Ratio, Latency, or Bandwidth? Have It All with Footprint Cache," in Proc. 40th Annual International Symposium on Computer Architecture (ISCA), 2013.

[4] N. Gulur, M. Mehendale, R. Manikantan, and R. Govindarajan, "Bi-Modal DRAM Cache: Improving Hit Rate, Hit Latency and Bandwidth," in Proc. 47th Iternational Symposium on Microarchitecture (MICRO), 2014.

[5] J. H. Ahn, S. Li, S. O, and N. P. Jouppi, "McSimA+: A Manycore Simulator with Application-level+ Simulation and Detailed Microarchitecture Modeling," in Proc. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2013.

[6] "CloudSuite benchmark 2.0." http://parsa.epfl.ch/cloudsuite.