

Gated Precharging: Using Temporal Locality of Subarrays to Save Deep-Submicron Cache Energy

Se-Hyun Yang, and Babak Falsafi

Computer Architecture Lab at Carnegie Mellon (CALCM)
Electrical and Computer Engineering Department
Carnegie Mellon University
{sehyun, babak}@ece.cmu.edu

Extended Abstract

Modern high-performance cache implementations use subarrays to reduce the capacitive load on the bitlines and achieve faster access time [6]. To overlap bitline precharging time with address decoding and wordline assertion, caches typically precharge all subarrays simultaneously prior to a cache access. Though only a small number of subarrays are accessed on a cache access, precharging all subarrays leads high energy consumption in modern and future high-performance deep-sub micron caches, because current CMOS scaling trends significantly increase the leakage from bitlines as process generation evolves [2, 3].

To precharge only the required subarrays (and reduce precharging activity and energy), *delayed precharging* technique was proposed, which uses part of address information to identify the subarray to be touched. [4, 5] Therefore delayed precharging ensures to precharge only as many subarrays as the set-associativity of the cache. However, the technique directly affects the cache access time by placing the precharge time on the cache access critical path. Our simulations on SPEC2000 and Olden benchmarks show that the performance impact of delayed precharging is over 10% for both level-1 instruction and data caches.

Recently, *resizable caches* have also been proposed. [1, 7, 8] Resizable caches identify the cache size requirements during or prior to the application execution and provide only the required cache size for each program phase or for the whole program execution. The unused cache portion is not precharged to save energy. However, to avoid block aliasing and maintain the correctness, data in the disabled subarrays are destructive and dirty data are written back to lower level caches on every cache resizing. This process increases cache misses and incurs performance impact. Also, resizable caches' resizing granularity is coarser than subarray size, resulting in sub-optimal offerings of cache sizes. [8]

In this research, instead, we propose *gated precharging* technique, which exploits cache accesses' temporal locality on cache subarrays to identify the subarrays that are most likely to be touched in near future. Our simulation results with SPEC2000 and Olden benchmarks show high temporal locality on cache subarrays for both data and instruction caches; 95% of level-1 data cache hits occur in the subarrays touched during the last 100 cycles, and instruction cache accesses are even more localized, so that we experienced 99% of level-1 instruction cache hits occurring in the subarrays touched during the last 100 cycles.

As a temporal locality measure, gated precharging associates a simple cycle-counter with each subarray. The counter is used to measure how many cycles it takes since the last touch on the subarray. Every access on a subarray resets the corresponding counter. When the counter value passes a certain threshold value, the subarray is not precharged until the next cache access on itself. We call the subarray in this status to be

disabled. As opposed to disabled subarrays, the subarrays that are touched recently and precharged on cache access are called *enabled*.

Cache hits on disabled subarrays are delayed by one extra cycle to precharge the subarray and enable the subarrays. With high temporal locality of the subarrays, the cache hits are more likely to occur on the enabled subarrays, and disabled subarrays stay long in the disabled status. Therefore, the precharging activity of caches is effectively reduced with minimal performance impact. In an 8-way out-of-order superscalar processor, simulation results on SPEC2000 and Olden benchmarks show that, within 2% of performance impact, gated precharging reduces 85% of the precharging activity of 32K direct-mapped level-1 data cache on average. For 32K direct-mapped instruction cache, more than 90% of the precharging activity is removed with only 2% of performance impact; only slightly more than one subarrays are enabled on average. Instruction cache shows better reduction because instruction footprints are typically more stable and exhibit higher temporal locality than data accesses patterns.

References

- [1] D. H. Albonesi. Selective cache ways: On-demand cache resource allocation. In *Proceedings of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 32)*, pages 248–259, Nov. 1999.
- [2] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, July 1999.
- [3] B. Davari, R. Dennard, and G. Shahidi. CMOS scaling for high performance and low power- the next ten years. *Proceedings of the IEEE*, 83(4):595, June 1995.
- [4] J. H. Edmondson, P. I. Rubinfeld, P. J. Bannon, B. J. Benschneider, D. Bernstein, R. W. Castelino, E. M. Cooper, D. E. Dever, D. R. Donchin, T. C. Fischer, A. K. Jain, S. Mehta, J. E. Meyer, R. P. Preston, V. Rajagopalan, C. Somanathan, S. A. Taylor, and G. M. Wolrich. Internal organization of the Alpha 21164, a 300-MHz 64-bit quad-issue CMOS RISC microprocessor. *Digital Technical Journal*, 7(1), 1995.
- [5] J. Montanaro, R. T. Witek, K. Anne, A. J. Black, E. M. Cooper, D. W. Dobberpuhl, P. M. Donahue, J. Eno, G. W. Hoepfner, D. Kruckemyer, T. H. Lee, P. C. M. Lin, L. Madden, D. Murray, M. H. Pearce, S. Santhanam, K. J. Snyder, R. Stephany, and S. C. Thierauf. A 160-MHz, 32-b, 0.5- μ m CMOS RISC microprocessor. *IEEE Journal of Solid-State Circuits*, 31(11):1703–1714, 1996.
- [6] S. J. E. Wilson and N. P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 93/5, Digital Equipment Corporation, Western Research Laboratory, July 1994.
- [7] S.-H. Yang, M. D. Powell, B. Falsafi, K. Roy, and T. N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *Proceedings of the Seventh IEEE Symposium on High-Performance Computer Architecture*, Jan. 2001.
- [8] S.-H. Yang, M. D. Powell, B. Falsafi, and T. N. Vijaykumar. Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay. In *Proceedings of the Eighth IEEE Symposium on High-Performance Computer Architecture*, pages 151–161, Feb. 2002.

