Energy-Efficient Cache Architecture for Multimedia Applications

Chia-Lin Yang, Chien-hao Lee, Hung-Wei Tseng

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
{yangc, r91017, r92022}@csie.ntu.edu.tw

Abstract

Power consumption is an important design issue of current embedded systems. It has been shown that the instruction cache accounts for a significant portion of the power dissipation of the whole chip. Data caches also consume a significant portion of total processor power for multimedia applications because they are data intensive. In this paper, we propose two mechanisms to reduce dynamic power consumption for both instruction and data caches. The HotSpot cache adds a small cache between the CPU and L1 instruction. It identifies frequently accessed instructions dynamically and stores them in the L0 cache. The software-controlled cache architecture improves the energy efficiency of the data cache by allocating data types in an application to different cache regions. On each access, only the allocated cache regions need to be activated. We find that on the average, the HotSpot cache and software-controlled cache can achieve 52% and 40% energy reduction on instruction and data caches, respectively. Both schemes incur little performance degradation.

1 HotSpot Cache

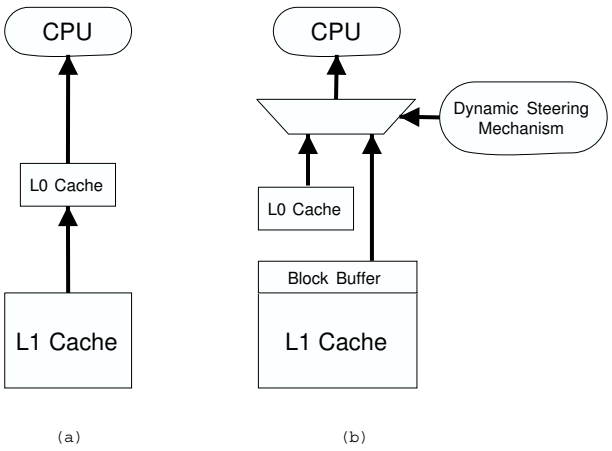

Figure 1: (a) Filter cache (b) HotSpot cache

Cache partitioning is commonly used to reduce the dynamic energy dissipation of caches since a smaller cache has a lower load capacitance. Block buffering [3] proposes to buffer the last accessed cache line. If the data in the same cache line is accessed on the next request, only the buffer needs to be accessed. A two-phase cache access scheme can be used to avoid performance degradation [4]. The Filter cache [2] adds a bigger buffer (i.e., the L0 cache) to cache recently accessed cache blocks as shown in Figure 1(a). On each access, the L0 cache is first accessed. The L1 cache is only accessed when an L0 miss occurs. This approach can achieve more energy reduction compared with the block buffering mechanism, however, it could cause significant performance degradation if an application's working set cannot be captured in the small L0 cache. Studies show that the performance degradation could be more than 20%. In this paper, we propose a novel instruction cache architecture, the HotSpot cache as shown in Figure 1(b). Unlike the Filter cache where the L0 and L1 cache are accessed sequentially, a dynamic steering mechanism is employed to direct a request to either the L0 or the L1 cache. The L1 cache is augmented with a block buffer. The design goal is to achieve energy savings comparable to the Filter cache with negligible performance degradation.

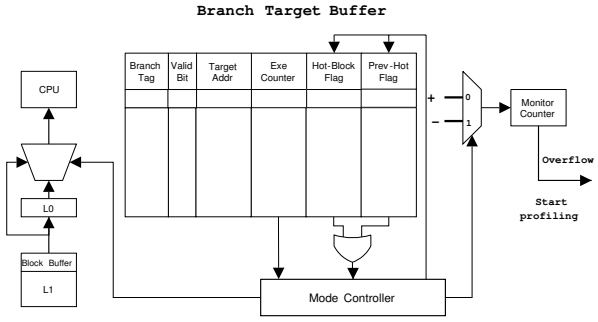

Figure 2: Block diagram of HotSpot cache

The HotSpot Cache identifies hot basic blocks in each program phase instead of the entire program lifetime. Bellas *et al.* [1] propose a static approach, L-Cache, that selects basic blocks to be mapped to the L0 cache based on profile information from the entire program execution. This approach may underutilize the L0 cache in program phases where identified hot basic blocks are not active. Therefore, we design a runtime mechanism that dynamically detects phase change and selects active hot basic blocks early in each program phase. To make such a hardware-based technique useful for low energy, we build the detection mechanism around the Branch Target Buffer. The block diagram of the proposed scheme is illustrated in Figure 2. The simulation results show that for applications with multiple phases, the HotSpot cache can achieve up to 2x higher L0 cache utilization than the L-Cache. Without the cost of performance degradation, the HotSpot cache achieves equal or more energy savings than the Filter cache for all applications tested in the paper. The energy reduction provided by the HotSpot cache is 52% on the average.

0-7803-9329-5/05/$20.00 ©2005 IEEE

2 Software-Controlled Cache

For an embedded system dedicated to a specific application, the cache architecture can be tuned to meet the need of that particular application. For an integrated multimedia system, a shared cache architecture among devices is often utilized to reduce hardware cost. Therefore, the conventional energy optimization technique which explores the design space in searching for the optimal energy-efficient cache architecture for a particular application can no longer be applied. The cache architecture can be only chosen to achieve energy optimization in an average sense.

Instead of tailoring the cache architecture to meet an application's need, in this paper, we propose a software-controlled cache mechanism that allows embedded software to manage the on-chip cache explicitly to improve the energy efficiency of the shared cache on an application-specific basis. The proposed mechanism allows a program to control data allocation on the cache in the granularity of data types in an application. Most multimedia applications contain several major data types that contribute to most of the memory accesses. The MPEG-2 software decoder, for an example, contains 6 data types which account for 80% of memory accesses according to our experiments. The proposed software-control cache architecture is a hardware/software cooperative scheme. The mapping between data types and cache regions is determined statically based on a programmer's knowledge on the application's behavior and profiling information. We derive the allocation methodology based on the optimization goal that achieves energy savings without sacrificing performance. The allocation information is then conveyed to the hardware through instruction annotations. The proposed software-controlled cache architecture only requires minimal hardware support, which is attractive to a low-power processor design.

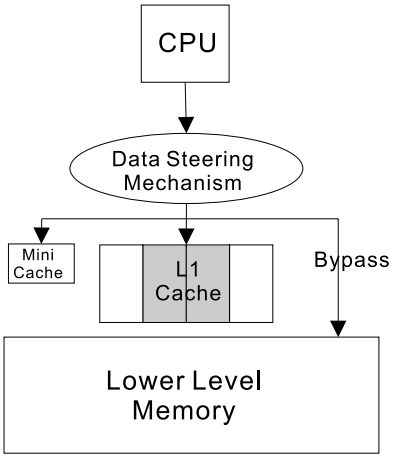

Figure 3: Software-controlled cache

The proposed software-controlled cache is designed based on an ARM-like cache architecture which contains a relatively small, directed-mapped cache (mini-cache) and a set-associative L1 cache on chip. Each way of the L1 cache is treated as a separate region. Each data type is either allocated to the mini-cache, ways of the L1 cache (way-partition) or bypasses the on-chip caches as shown in Figure 3. Bypassing memory references that have little reuse has been used to improve the L1 cache performance because it results in less cache pollution [5]. The cache bypassing technique can also be adopted for energy optimization. Most multimedia applications have a significant amount of output data that are written and never read by the processor. Bypassing these write-only data can reduce energy consumed in the memory subsystem for two reasons. First, it reduces off-chip accesses as a result of less cache pollution. Second, it reduces accesses to the L1 cache without increasing accesses to the lower level of the memory hierarchy since written data is never accessed again by the processor.[1] The results show that the software-controlled cache reduces energy consumption of an ARM-like cache by 40% without performance degradation.

3 Conclusion

In this paper, we propose two approaches to reduce cache power consumption: the HotSpot cache and software-controlled cache. The HotSpot cache is an architectural approach to dynamically select basic blocks for placement in the L0 cache. We design a mechanism that can successfully identify frequently accessed basic blocks in each program phase at runtime. Only basic blocks declared as hot blocks are stored in the L0 cache. The L1 cache is augmented with a block buffer for exploiting spatial locality within a cache line for energy savings. The simulation results show that the proposed mechanism can achieve more energy savings than the Filter cache. The energy consumption of the instruction cache is reduced by 52% with negligible performance degradation. The software-controlled cache allocates data types in an application to different cache regions. A data type is either mapped to the mini-cache, ways of the L1 cache or bypass on-chip caches. The optimization goal is to achieve energy reduction without performance degradation. We test the effectiveness of the software-controlled cache on the MPEG-2 software decoder. The results show that the software-controlled cache reduces 40% of energy on an ARM-like cache architecture without sacrificing performance.

References

[1] N. E. Bellas, I. N. Hajj, C. D. Polychronopoulos, and G. Stamoulis. Architectural and compiler techniques for energy reduction in high-performance microprocessors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3), June 2000.

[2] J. Kin, M. Gupta, and W. H. Mangione-Simith. The filter cache: An energy efficient memory structure. In *Proceedings of 30th Annual International Symposium on Microarchitecture*, December 1997.

[3] C.-L. Su and A. Despain. Cache design tradeoffs for power and performance optimization: A case study. In *Proceedings of International Symposium on Low Power Design*, April 1995.

[4] M. B. Kamble and K. Ghose. Analytical energy dissipation models for low power caches. In *Proc. of the International Symposium on Low Power Electronics and Design*, 1997.

[5] Teresa L. Johnson and Daniel A. Connors and Matthew C. Merten and Wen-mei W. Hwu Run-Time Cache Bypassing", journal = "IEEE Transactions on Computers In *IEEE Transactions on Computers*, 48(12),December, 1999.

[1] We assume a write-buffer between the L1 cache and the next level of the memory hierarchy to merge writes to the same cache blocks.