Power-aware deterministic block allocation for low-power way-selective cache structure

Jung-Wook,Park†, Gi-Ho,Park‡, Sung-Bae,Park‡, Shin-Dug,Kim†
† CS, Yonsei University 134, Shinchon-dong Seoul, 120-749, Korea +82-2-2123-2718
‡ Processor Architecture Lab. System LSI Division,Samsung Electronics Co., LTD. Giheung, Korea
pjppp@parallel.yonsei.ac.kr, {giho.park, sung.park}@samsung.co.kr, sdkim@yonsei.ac.kr

Abstract

This paper proposes a power-aware cache block allocation algorithm for the way-selective set-associative cache on embedded systems to reduce energy consumption without additional delay or performance degradation. For this goal, way selection logic and specialized replacement policy are designed to enable only one way of set-associative cache as in the direct-mapped cache. Overall cache access time becomes almost the same as that of conventional set associative cache with accessing additional way selection logic. Because data array can be accessed without waiting for tag comparison, multiplexer delay can be removed totally. The simulation result shows that the proposed architecture can reduce a per access power consumption by 59% over conventional set-associative caches with average 0.06% of negligible performance loss.

1. Introduction

As mobile and portable devices are widely used, recent studies about microprocessors concentrate on overall energy efficiency rather than performance because of both limitation on battery capacity and significant thermal problems due to the increased complexity and operation frequency of the system. In the design of embedded systems, memory subsystems such as cache and TLB may cause a direct impact on overall system performance and consume nearly half of the entire processor energy [15]. Even though static power becomes a major portion of whole processor power consumption as the advances in process technology, dynamic power reduction is still an important factor in low power design techniques. There are numerous architectural approaches to improving energy efficiency of cache memories by reducing dynamic cache access power, e.g., buffering, filtering mechanisms, and selective/reconfigurable associative caches. Direct mapped cache consumes least power for each access to a cache block, but it doesn't consume least energy comparing with many other approaches. This is because a cache miss causes an access to the lower level memory and those accesses consume more energy. Embedded systems for a specific application or domain need only simple cache memory system yet. But convergence of various applications requires more complex cache systems for their performance requirement.

There are two general approaches to reduce energy consumption of cache memories related to the cache associativities. The first one is to improve the performance of the direct mapped cache with simple additional mechanism. A representative architecture for this type of method is the victim cache. The victim cache dramatically reduces conflict misses, but there is also additional power consumption caused by accessing fully associative buffer simultaneously. The second one is to reduce the accessing power of set associative caches. Specially increasing the set-associativity causes performance improvement and additional delay time in determining a specific word among the outputs from every way. And also additional power consumption can be caused from accessing every cache way in parallel. Analysis of cache power dissipation [4] shows that data bitlines and data sense amplifiers are responsible for 55% (direct mapped), 65% (2-way) and 75% (4-way) of the power consumption for a given cache structure. According to this analysis, various selective way (associativity) accessing mechanisms have been proposed, e.g., phased-access cache, way-prediction cache [3], and predictive sequential cache. Decreased accessing power of the phased-access cache tends to cause intolerable latency overhead. The way prediction mechanism is more desirable, but its efficiency depends on the accuracy of the prediction strategy because of its

expensive miss prediction penalty. To overcome this kind of overheads in performance and power, several studies about minimizing delay incurring for sequentially accessing steps are performed.

In processor design, cache memory system is one of well known timing critical paths. Thus it is difficult to use additional hardware which increases access time, even though it achieves low power consumption. We propose an effective and realizable way-selecting structure to eliminate timing problems in previous approaches. In addition to fast access time, we can reduce power consumption via parallel tag access. Because tag access path is more likely to be a critical path of cache access time and tag decoding time is much shorter than that of data path, any kind of improved circuit techniques cannot reduce power consumption of tag part. Proposed cache structure solves this problem by providing only one output at each cache access. This mechanism changes conventional critical path which includes tag path with way selection and data array access and also removes multiplexer (MUX) driver delay in critical path. Eventually, overall cache access time becomes almost the same as that of conventional set associative cache. Our analysis using Cacti power model shows that the proposed cache structure reduces per access power consumption by 59 % of conventional set-associative cache structure.

2. Related work

Phased cache looks up tag arrays and data array sequentially. There is no wasted energy to access unmatched data sub-array. But, all of the load instructions are delayed by one more cycle. This latency significantly degrades the overall performance.

Way-prediction cache [3] speculatively chooses one set before cache line access in the set associative cache. MRU (Most Recently Used) bits of each entry have information of recent accesses to select the first way to access. If prediction is correct, cache consumes energy for only one activated way. Otherwise, the cache searches all of the ways and consumes energy for all of them. In addition to this energy consumption, miss prediction also causes additional cycles that can degrade performance. Moreover non-deterministic cache access delay due to miss-prediction can cause significant overhead in whole pipeline implementation. This additional circuit may also consume dynamic and static power.

Adaptive serial-parallel CAM cache [10] can sequentially access tag array. In serial mode for low power consumption, the least significant four bits of each row are checked serially and remaining bits are checked in parallel on the next cycle only if serial check was matched. This cache structure has flexibility that it can be also operated as conventional fast parallel CAM cache for better performance without power reduction.

Way-halting cache [11] also uses fully associative memory for four bits wide way-halting logic. It uses static CAM array to check the least significant four bits of tag in parallel to data address decoding. Access to mismatched way of data array would be halted by the wordline gating mechanism. This approach shows that static CAM structure has shorter latency than conventional index decoding time, and static CAM structure does not increase overall energy consumption significantly because tag bits are changed scarcely particularly for instruction cache. However this mechanism is not well adapted to data cache as authors mentioned in their paper. This structure can consume increased energy for data cache in way-halting static CAM arrays because data cache access patterns are more complicated and less predictable which means frequent change in contents of static arrays. Moreover this cache structure has difficulty in scaling the size of cache because CAM structure has small threshold size that cannot cause significant increase of power consumption due to broadcasting of comparing bits for every entry.

When we use several bits of tag address to determine the way to be accessed, prediction accuracy can be improved by using the least significant bits of tag address. This is based on the spatial locality of data access pattern [11]. Our approach utilizes the same characteristics of data locality in a different way. Specifically, there may be low possibility that data blocks stored in the same set have the same low order tag bits. In turn, this property can be interpreted in the way that data blocks with the same low order bits of tag address do not tend to cause conflict misses. In the proposed architecture, we need some constraints on replacement policy to obtain low power consumption. Further details about the effects of this algorithm for power consumption and delay latency will be described in the following sections.

3. Single-way selective cache

We propose a new single-way selective (SWS) cache for low power consumption without any additional accessing latency via specialized replacement algorithm. As mentioned before, this replacement mechanism allocates the blocks which

have different least significant four bits of tag address in a set. In this section, the proposed cache organization and operational model are described in detail.

3.1. Cache organization

The basic SWS cache model is designed as a four-way set-associative cache. In addition to the conventional set-associative cache structure, the SWS cache has arrays of four-bit mini-tag composed of low order four bits of the tag address in each cache way. This array can be implemented in the same way as the CAM array [11] or as the conventional RAM array and comparators. In our approach, it is possible to remove MUX driver which selects outputs from every cache way to overcome additional delay accompanied by sequential way selecting process because only one way can be enabled on each access.

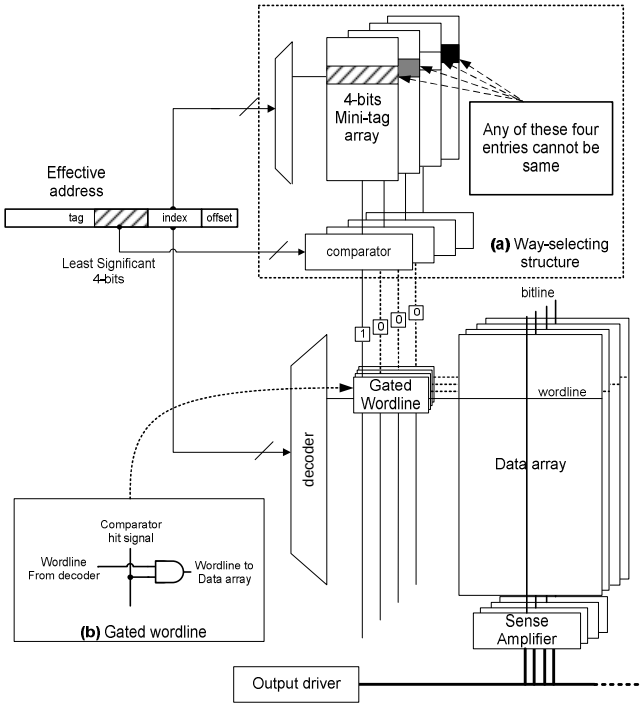

Figure 1. SWS cache organization

3.2. Operational model of the SWS cache system

Here, we describe the algorithm for managing the SWS cache in detail. Its conceptual operation flow is explained as follows. When a cache miss occurs, an array of four entries with four-bit-wide tag for way selection should be compared with those of requested address. If a match occurs in a particular way, a new block from the lower level memory is fetched into that way to guarantee one way access. Otherwise a new block can be placed into any set according to conventional replacement policy. And the replaced block will be evicted from the entire cache. Different cases for the operational model are explained as follows.

3.2.1. Cache access

When a load/store unit generates a data address, accessing the data, tag and way-selecting sub-array is started using the index field bits of the data address. Only one wordline of data array will be activated through gated wordline (b part in Figure 1) of the result of mini-tag comparison (a part in Figure 1). Because the proposed cache system has constraints that data blocks in each way at the same set cannot have the same way-selecting bits, only one wordline is enabled.

3.2.2. Victim block select

When a miss occurs, the location for a new data block will be determined by simple power-aware replacement algorithm. To avoid same blocks reside among the four entries in an index set of mini-tag array, compare the four entries with least significant tag bits of requested address which causes a cache miss. If there is a match in the way-selecting sub-array, the matching block will be a victim block. Otherwise, conventional replacement algorithm such as LRU replacement algorithm can be used. In general, LRU strategy can show the best performance in conventional set-associative cache but we use random or round-robin replacement policy for the proposed system because proposed replacement algorithm can decrease the effectiveness for the LRU replacement algorithm. According to the simulation result that will be described in later section, random or round-robin replacement policy can show better performance than the LRU with proposed way-selective cache.

4. Delay and power analysis

In this section, we describe detailed analysis on delay and power consumption for each component of the proposed cache system to compare with conventional set-associative cache structure. We calculate all the values in this section through CACTI 3.0 power model and tools for 0.13 ㎛ technology.

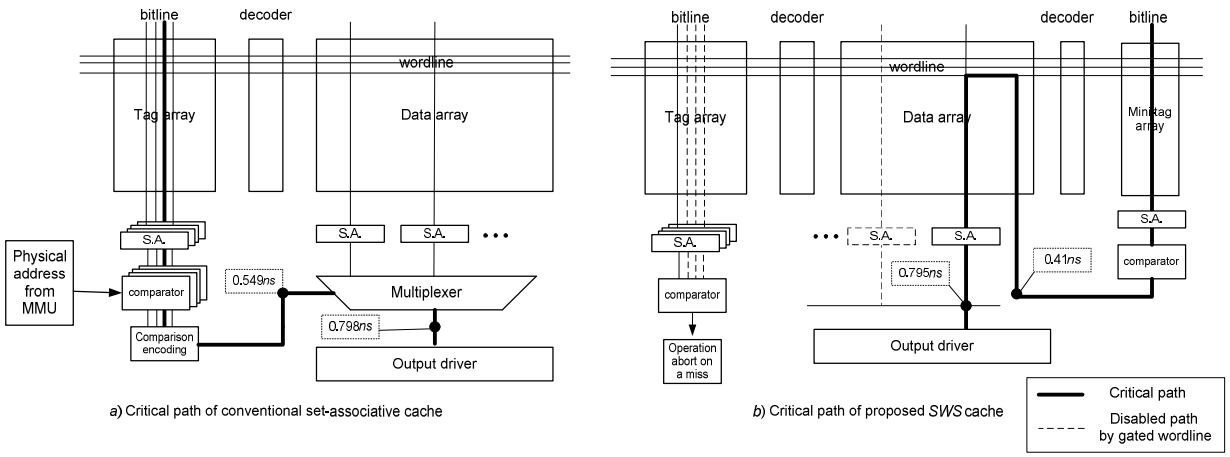

Figure 2. Access time scenario model with critical path delay

4.1. Cache access time in conventional set-associative cache

In most of cache designs, path for tag comparison logic is critical path of overall cache access. Critical path delay for whole structure of conventional set-associative cache and each component with cumulative result of delay time can be described as follows.

$$T_{cache\_access} = T_{tag\_index\_decode}(0.156ns) + T_{tag\_array\_w/b\_line}(0.343ns) + T_{tag\_comparison}(0.549ns) + T_{MUX\_driver}(0.798ns) + T_{output\_driver}(0.919ns)$$

In addition to tag array access, most cache structures perform TLB access in parallel to cache access and tag value should be compared with physical address from TLB. For that reasons, delay time for the TLB access also can lengthen the critical path delay for tag comparison. This delay time is one of the most important factor of cache access latency. ***a)*** of Figure 2 shows timing critical delay path of conventional set-associative cache structure.

4.2. Cache access time in proposed SWS cache

The main source of delay increasing the overall access time can be the one for accessing the way-selecting sub-array. In conventional set-associative caches, all the sets at the same index are accessed simultaneously to detect a match. However, a particular way is determined by processing serially in this proposed approach. These serial accessing characteristics can be an important factor to utilize in designing low power systems. Most of parts in cache architecture are involved in the critical path of processor cycle time. Particularly, there is no enough timing margin between the time to access cache and the time to generate an effective address in load/store unit. So we cannot ignore the additional delay of sequential access in the proposed cache memory system. The proposed architecture eliminates tag path delay from the critical path by at most single output for every cache accesses as shown in the ***b)*** of Figure 2. Changed critical path delay for whole SWS cache structure and its components with cumulative result of delay time can be described as follows.

$$T_{SWS\_cache\_access} = T_{mini\_tag\_index\_decode}(0.11ns) + T_{mini\text{-}tag\_w/b\_line}(0.25ns) + T_{mini\_tag\_comparison}(0.41ns) + T_{data\_w/b\_line}(0.795ns) + T_{output\_driver}(0.91ns)$$

Moreover delay time of MUX driver which selects a correct output data among the multiple ways and spends much time, can be eliminated.

4.3. Power consumption

Four bit comparisons and their accesses to the array in the proposed structure consume only 6% of total cache power consumption. Power reduction can be achieved by 68% for data sub-arrays. In addition to power reduction from the data sub-array, proposed cache structure can achieve additional power reduction from way-selection on tag array because path for tag compare logic does not affect any impact on cache access time in changed critical path. Previous research about direct-mapped cache [12] shows that if only one data block is available, data can be fetched to the CPU before the outcome of the tag check is known. And also processing of the instruction is aborted at a later stage in the pipeline when a miss occurs. This feature in the SWS cache can achieve more power reduction by 58% for tag sub-arrays. To take additional hardware and all power reduction into account, total power consumption can be reduced by 59% per an access to the cache of the conventional set-associative cache.

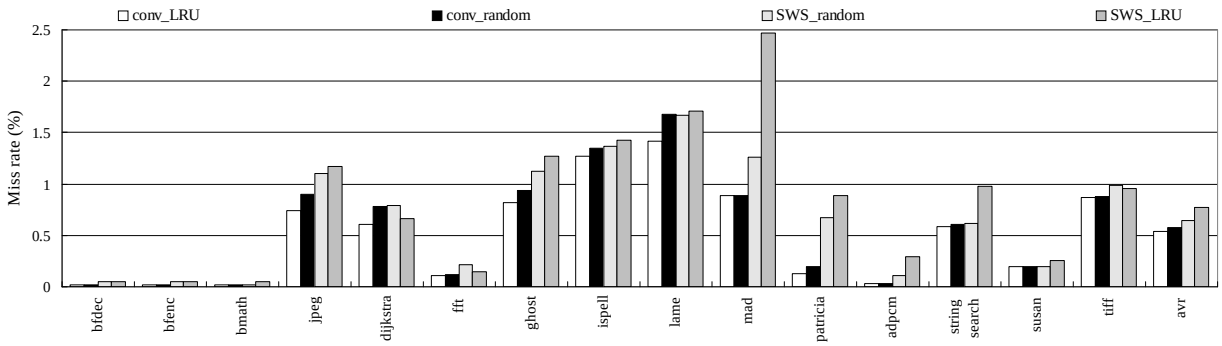

Figure 3. Cache miss rates for various embedded applications.

5. Results

The proposed SWS cache is based on the assumption that changing the replacement algorithm will not increase cache miss ratio significantly. We verify the assumption through the simulation result of various embedded applications and analyze overall power consumption by the result of simulation.

5.1. Experimental methodology

We use SimpleScalar/ARM processor simulator to gather statistics of the proposed cache architecture. Simulated target machine is configured as the settings of representative embedded processor as parameter values shown in Table 1.

Table 1. Simulation target machine configuration.

Execution order	In-order
L1 data cache	16KB, 32byte, 4way
L2 cache	None
TLB entry/configuration	64 / fully associative
Main memory access latency	22 cycle (13+3+3+3)

5.2. Simulation results

We use Mibench benchmark suit [13] which represents various domains of embedded workloads. We compared four kinds of cache memory systems. First, denoted as 'conv_LRU', 'conv_random' in Figure 3 is configured as a 16KB size, 32B block, 4-way set-associative cache with LRU, random replacement policy and the others are the proposed way-selective caches denoted as 'SWS_random' and 'SWS_LRU'. We omit results of miss rate for some benchmarks which have very small miss ratio at every four cache structures. Cache performance result in Figure 3 shows interesting fact that the proposed four bits constraints brings worse performance with LRU rather than random policy. On the average, the replacement algorithm for way selection logic increases cache miss rate for 0.1%, 0.2% with random, LRU replacement policy respectively. The degradation of the miss ratio does not cause the same amount of performance. According to the performance simulation results that are not shown in this paper because the difference among the result values is too small to be recognized on the graph, overall performance loss is up to 0.59% for the Mad application with LRU, on average 0.06%, 0.13% with random and LRU respectively. For basic model of the proposed cache architecture, we achieve 59% per access power reduction. Overall energy consumption of proposed cache system can be calculated by analytical method presented in previous work [11],[14] as follows.

$$overall\_energy = number\_of\_cache\_hits * hit\_energy + number\_of\_misses * miss\_handling\_energy$$

According to the analysis for power model, the power consumption caused by cache miss handling is around 50 to 200 times larger than per access power. Overall energy consumption of various cache structure through the power model with 50 of miss power factor, is shown in Figure 4. The result with power factor of 200 brings similar result. Without plotting the result of '200', Figure 4 shows energy consumption of each cache structure for cache access and miss handling separately. By the result, our cache architecture shows that average energy reduction is calculated as 56% down to 38% for each case.

6. Conclusion

Generally, it is difficult to put additional hardware logic for reducing the cache power consumption because most of cache access time may be involved in the critical path of processor cycle time. We propose the SWS cache exploiting the way-prediction to

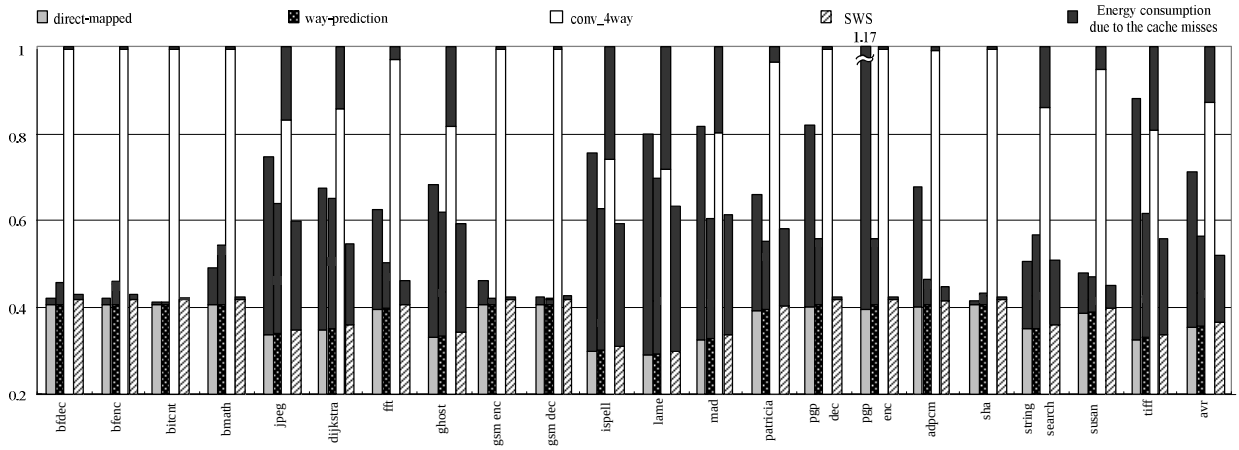

Figure 4. Overall energy consumptions normalized to conventional 4-way cache

achieve low power consumption without additional delay. All of this energy efficiency can be obtained when cache accesses with additional delay can be performed within a cycle.

Marginal time to access the way selection logic can be obtained by removing tag path delay from the critical path by at most single output for every cache accesses and eliminating the multiplexer delay in the critical path. Therefore the overall cache access time of the proposed cache system becomes almost the same or even faster than that of conventional set associative cache.

7. REFERENCES

[1] B.Calder, D.Grunwald, and J.Emer, "Predictive sequenti al associative cache,"Proceedings of Second International Symposium on High-Performance Computer Architecture, Feb. 1996, pp.244 -253.

[2] D.H.Albonesi, "Selective cache ways: on-demand cache resource allocation," Proceedings of 32nd Annual International Symposium on Microarchitecture. Nov. 1999, pp.248 -259.

[3] K.Inoue, T.Ishihara, and K.Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," Proceedings of International Symposium on Low Power Electronics and Design, Aug. 1999, pp.273 -275.

[4] S.J.E.Wilton and N.P.Jouppi, "CACTI: an enhanced cache access and cycle time model," IEEE Journal of Solid-State Circuits, Volume: 31 Issue: 5 , May 1996, pp.677 -688.

[5] M.D.Powell, A.Agarwall, T.N.Vijaykumar, B.Falsafi, and K.Roy, "Reducing set-associative cache energy via way-prediction and selective direct-mapping," Proceed ings of 34th ACM/IEEE International Symposium on Microarchitecture, MICRO-34. Dec. 2001, pp.54 -65.

[6] J.H.Lee, S.W.Jeong, S.D.Kim, and C.Weems, "An Intelligent Cache System with Hardware prefetching for High Performance," IEEE Transaction on Computers , Vol. 52, No. 5, May 2003, pp. 607-616.

[7] N.P.Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," Proceedings of 17th Annual Internatio nal Symposium on Computer Architecture, May 1990

[8] D.Burger, T.M.Austin, "The SimpleScalar tool set, version 2.0," Technical Report TR-97-1342, University of Wisconsin-Madison, 1997.

[9] S.Kim, N.Vijaykrishnan, M.J.Irwin and L.K.John "On load latency in low-power caches", Proceedings of International Symposium on Low Power Electronics and Design, Aug. 2003 pp.258-261

[10] A.Efthymiou, J.D.Garside "An adaptive serial-parallel CAM architecture for low-power cache blocks", Proceedings of International Symposium on Low Power Electronics and Design, Aug. 2002 pp.136-141

[11] C.Zhang, F.Vahid, J.Yang, W.Najjar "A Way-Halting Cache for Low-Energy High-Performance Systems" IEEE Computer Architecture Letters, Vol. 2, 2003

[12] D.Stiliadis, A.Varma "Selective victim caching: A method to improve the performance of direct-mapped caches", IEEE Transaction on Computers , Vol. 46, No 5, May 1997, pp. 603-610

[13] M.R.Guthaus, J.S.Ringenberg, D.Ernst, T.M.Austin, T.Mudge, R.B.Brown "MiBench: A free, commercially representative embedded benchmark suite", Proceedings of IEEE International Workshop on Workload Characterization, Dec. 2001,pp 3-14

[14] C.Zhang, F.Vahid, and W.Najjar, "A highly configurable cache architecture for embedded systems," Proceedings of 30th Annual International Symposium on Computer Architecture, June. 2003, pp.136 -146.

[15] S.Segars, "Low power design techniques for microprocessor" Tutorial, International Solid-State Circuit Conference, Feb. 2001.