A Single Cycle Accessible Two-Level Cache Architecture for Reducing the Energy Consumption of Embedded Systems

Yamaguchi, Seiichiro
Graduate School of Information Science and Electrical Engineering, Kyushu University

Ishihara, Tohru
System LSI Research Center, Kyushu University

Yasuura, Hiroto
Faculty of Information Science and Electrical Engineering, Kyushu University

出版情報：Proceedings of International SoC Design Conference. 2008, pp.188-191, 2008-11-25
バージョン：
権利関係：

A Single Cycle Accessible Two-Level Cache Architecture for Reducing the Energy Consumption of Embedded Systems

Seiichiro Yamaguchi
Graduate School of ISEE
Kyushu University, Japan
seiichiro@c.csce.kyushu-u.ac.jp

Tohru Ishihara
System LSI Research Center
Kyushu University, Japan
ishihara@slrc.kyushu-u.ac.jp

Hiroto Yasuura
Faculty of ISEE
Kyushu University, Japan
yasuura@c.csce.kyushu-u.ac.jp

Abstract— Employing a small L0-cache between an MPU core and an L1-cache is one of the most promising approaches for reducing the energy consumption of memory subsystems. Since the L0-cache is small, if there is a hit, the energy consumption will be reduced. On the other hand, if there is a miss, one extra cycle is required to access the L1-cache. This leads to a degradation of the processor performance. For resolving this problem, a Single cycle accessible Two-level Cache (STC) architecture is proposed in this paper. This architecture makes it possible to access to both the L0 and the L1 caches from an MPU core in a cycle. Experiments using several benchmark programs demonstrate that the STC architecture reduces the energy consumption of memory subsystems by 13% without any performance degradation compared to the best results obtained by previous approaches.

Keywords: Cache memory, Low power design

I. Introduction

Reducing the energy consumption is one of the most important criteria for microprocessors. They also require ever-increasing performance for integrating multiple functionalities into a single system. Today's microprocessors have on-chip caches in order to improve the performance. The on-chip caches also improve the energy efficiency of memory subsystems through decreasing the total number of accesses to off-chip memories which involve huge energy dissipation. However integrating too large caches on a chip results in an increase of the total energy consumption since the energy consumption of the cache becomes dominant as the size of the on-chip cache increases. For example, ARM920T microprocessor dissipates 44% of the power in its caches [5]. StrongARM SA-110 microprocessor, which specifically targets low-power applications, dissipates 43% of the power in its caches [3]. Therefore, a cache architecture which reduces the total number of off-chip accesses without increasing the energy consumption of the cache is highly desired. In this paper, a new cache architecture which reduces the energy consumption of the cache and the total number of off-chip accesses is proposed.

The proposed cache architecture, named a Single cycle accessible Two-level Cache (STC) architecture, has one small and one normal size caches at the same level of memory hierarchy. Only one of the caches is activated at a time. Since both of them can be accessed from an MPU within a cycle, there is no performance degradation compared to the conventional level-1 caches. If the small cache is more frequently accessed, the total cache energy consumption can be reduced. Although the two caches are located at the same level of cache hierarchy, they also has a property of hierarchical caches. Frequently accessed code and data are placed in a memory address space so that they are mapped to the small cache preferentially. This concentrates memory accesses to the small cache and, as a result, reduces the total energy consumption of caches.

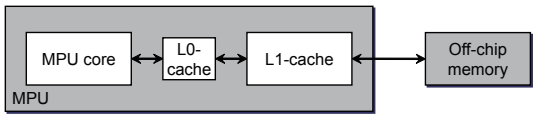

Figure 1. Architecture of MPU with L0-cache.

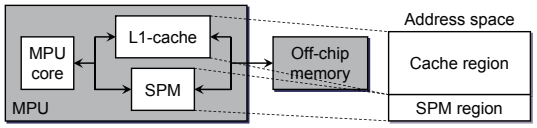

Figure 2. SPM-based architecture and its address mapping.

The rest of this paper is organized as follows. In section II, we describe related work. Section III presents a motivational example and our approach. The detailed STC architecture is shown in section IV. Section V shows experimental results. Finally, section VI concludes this paper.

II. Related Work

In the past, many researchers have proposed a technique exploiting a small L0-cache between an MPU core and an L1-cache, e.g., S-Cache [4] and Block Buffering [1]. Figure 1 shows an architecture of a processor with the L0-cache. Since the L0-cache is small, it consumes less energy per access. Therefore, if there is a hit in the L0-cache, the energy consumption will be reduced. On the other hand, if there is a miss, one extra cycle is required to access the L1-cache. This results in a degradation of the microprocessor performance.

A software controlled memory called scratchpad memory (SPM) is used with an L1-cache for resolving the problem of the L0-cache. Since the SPM and the L1-cache are located at the same level of memory hierarchy, both of them can be accessed from an MPU core in a single cycle. Only one of them is activated at a time. Figure 2 shows an SPM-based architecture and its address mapping. The code and data allocation to the SPM is typically done during a system boot and is unchanged after the boot. The SPM consumes less energy per access compared to that of the L1-cache since the SPM does not need tag

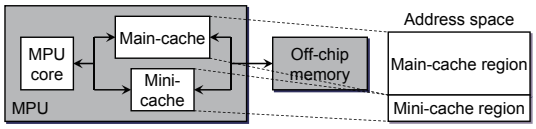

Figure 3. HPC architecture and its address mapping.

TABLE I. MOTIVATIONAL EXAMPLE

Code	Access ratio	Size
Subroutine A	0.4	1KB
Subroutine B	0.4	1KB
Others	0.2	30KB

Cache	Size
Main-cache	8KB
Mini-cache	1KB

Case	Code allocation	
	Mini-cache region	Main-cache region
1	Subroutine A, Subroutine B	Others
2	Subroutine A	Subroutine B, Others
3	Subroutine B	Subroutine A, Others

search operations which are needed for caches. Therefore, the SPM is more energy efficient if programmers or compilers can optimally allocate code and data to the SPM [7].

Horizontally Partitioned Cache (HPC) architecture exploits two caches, a small and a normal size caches, at the same level of memory hierarchy [2, 6]. The HPC architecture and its address mapping are shown in Figure 3. Main-cache and Mini-cache respectively represent a normal size cache and a small cache. The address space is partitioned into two regions, Main-cache region and Mini-cache region. These regions are exclusively mapped to the Main-cache and the Mini-cache respectively. After checking several bits of a memory access address, an MPU core accesses to one of the caches. A memory module which is not accessed is not activated. This mechanism is similar to the memory subsystem with the SPM architecture. The energy consumption of the memory subsystem is also reduced by allocating frequently accessed code and data to the Mini-cache region.

III. MOTIVATION AND OUR APPROACH

One of the major issues in the HPC architecture is its ineffective utilization of on-chip memory resources. For example in the HPC architecture, the code and data allocated to the Mini-cache region are mapped to the Mini-cache only. Therefore, they cannot exploit large capacity of the Main-cache. On the other hand, those allocated to the Main-cache region cannot exploit energy efficiency of the Mini-cache.

More specific example is described in TABLE I. In this example, subroutines A, B and the others are alternatively executed on the HPC architecture. If these subroutines are less frequently switched as shown in Figure 4 (a), code allocation case 1 in TABLE I is better allocation than cases 2 and 3 with respect to the energy consumption since the number of Mini-cache misses is not very large. Contrarily, if the subroutines are more frequently switched as shown in Figure 4 (b), the code allocation case 1 results in an increase of the number of Mini-cache misses. For reducing the number of Mini-cache misses, we can allocate one of the subroutines to the Main-cache region as shown in cases 2 and 3. However, this increases the

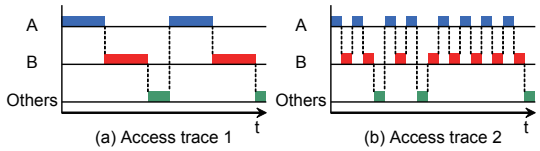

Figure 4. Access trace examples.

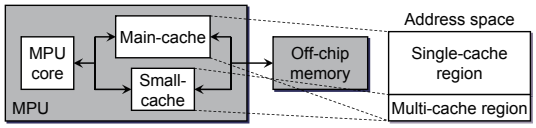

Figure 5. STC architecture and its address mapping.

average energy required for accessing caches. We can resolve the problems in the HPC architecture if the subroutines A and B allocated to the Mini-cache region can be also mapped to the normal size cache. Our approach allows the code and data allocated to the Mini-cache region to be mapped to both the small and the normal size caches.

Our STC architecture employs a small cache called Small-cache as is employed in the L0-cache and the HPC architectures. Figure 5 shows a memory subsystem with the STC architecture and its address mapping. The address space is partitioned into two regions, a Multi-cache region and a Single-cache region. The difference between the HPC architecture and the STC architecture is an address map of the Multi-cache region. In our STC architecture, the Multi-cache region is mapped to both the Small-cache and a Main-cache. The Single-cache region is mapped to the Main-cache only. The code and data allocated to the Multi-cache region exploit the Small-cache preferentially, and utilize the Main-cache depending on a replacement policy.

IV. STC ARCHITECTURE

In our STC architecture, only one of the Main-cache or the Small-cache is activated at a time. This is done by checking several bits of the memory access address. Since the Single-cache region is mapped to the Main-cache only, only the Main-cache is activated if the target address is included in the Single-cache region. However, since the Multi-cache region can be mapped to both the Small-cache and the Main-cache, it is impossible to decide which caches should be activated before accessing a tag of the Small-cache if the target address is included in the Multi-cache region. Accessing both the Small-cache and the Main-cache in parallel leads to an increase of the energy consumption of the caches. Accessing the Small-cache first and then accessing the Main-cache if there is a miss in the Small-cache reduces the total energy consumption of the caches. However, this degrades the processor performance. To avoid these issues, the STC architecture employs a mechanism to detect Small-cache hits and misses before activating the caches. This mechanism can be implemented by a flip-flop-based tag memory. If the size of the Multi-cache region is N times larger than the Small-cache size, $\lceil log_2(N) \rceil$-bit is needed for implementing the tag memory. The tag search oper-

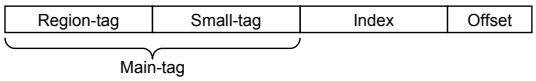

Figure 6. Memory access address of the STC architecture.

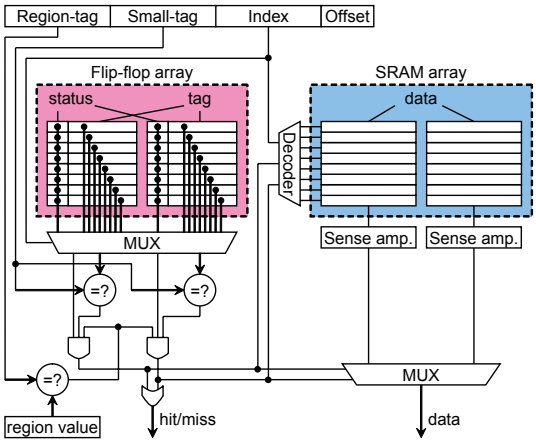

Figure 7. 2-way Small-cache architecture.

ation for the Small-cache is performed in parallel with checking whether the target address resides in the Single-cache region or the Multi-cache region. The Small-cache is activated if code or data is allocated to the Multi-cache region and if there is a Small-cache hit, otherwise the Main-cache is activated.

The memory access address consists of Region-tag, Small-tag, Index, and Offset fields as shown in Figure 6. The Region-tag field specifies whether the access address resides in the Single-cache region or the Multi-cache region. The Small-tag field is used for detecting a Small-cache hit or a miss. The Main-tag field which consists of the Region-tag and the Small-tag fields is used for checking a Main-cache hit or a miss. Figure 7 shows an architecture of the Small-cache. Status and tag fields of the Small-cache consist of a flip-flop array for checking a Small-cache hit or a miss before activating word-lines of caches. Read and write accesses to an SRAM array are done in the following steps, 1) decode a row address and precharge bit-lines, 2) activate one of word-lines, and 3) read out data by a sense amplifier. Meanwhile, a value in the flip-flop array can be obtained quickly since, unlike an SRAM module, an output signal of the flip-flop is always available without any read out operations. Hence, the delay for checking a Small-cache hit or a miss depends on delays required for a multiplexer and a comparator. The procedure for checking a Small-cache miss has to be completed before activating a word-line of the Main-cache. Otherwise, an access to the Main-cache needs more than one cycle. Since the tag search operation of the Small-cache in our STC architecture is performed every cycles, the average energy consumption per a Small-cache access is larger than that of the HPC architecture. A read and write flows for caches are described below.

Read and write flow

RW1 Region-tag comparison
: Check whether or not the target address is included in the Single-cache region. If it is in the Single-cache region, go to the step RW2. Otherwise, go to the step RW3.

RW2 Main-tag comparison
: If there is a hit, perform read or write operation for the Main-cache. Otherwise, get the target line from an off-chip memory and overwrite a line having the lowest priority in the target set of the Main-cache.

RW3 Small-tag comparison
: If there is a hit, perform read or write operation for the Small-cache. Otherwise, go to the step RW4. This comparison is operated in parallel with the step RW1.

RW4 Main-tag comparison
: If there is a hit, perform read or write operation for the Main-cache and store the operated line to a Small-cache replacement buffer. Otherwise, get the target line from an off-chip memory and store the line to the Small-cache replacement buffer. Go to the Small-cache replacement flow.

Small-cache replacement flow

SR1 Make backup copy
: Make a copy of data in a line having the lowest priority in the target set of the Small-cache and store it to a Main-cache replacement buffer. Go to the next step. Note that the lowest priority line can be determined based on replacement policies like the least recently used (LRU) policy or the least frequently used (LFU) policy. This step is operated in parallel with the step RW3 in case of a miss.

SR2 Update Small-cache
: Move the data in the Small-cache replacement buffer to the lowest priority line of the Small-cache. Update the Small-tag simultaneously. Go to the next step.

SR3 Update Main-cache
: Move the data in the Main-cache replacement buffer to the lowest priority line of the Main-cache. Update the Main-tag simultaneously. This step is operated in parallel with the step SR2.

An invalid flag of a cache line is set when the line resided in the Small-cache is replaced to the Main-cache or the line resided in the Main-cache is copied to the Small-cache. This preserves coherence between the Main-cache and the Small-cache. For always achieving the single cycle access to both the Small-cache and the Main-cache, the Small-cache replacement procedure is not performed if the Small-cache replacement buffer or the Main-cache replacement buffer is full. Otherwise, one extra cycle is needed for waiting for completing the Small-cache replacement procedure before accessing them.

V. Experimental Results

A. Experimental Setup

According to our original SRAM modules designed using a commercial 65nm technology, the energy consumptions required for accessing to the cache modules in the HPC architecture are as shown in TABLE II. Note that the Mini-cache and the Main-cache used in our experiments are 1KB direct-map and 8KB 4-way set-associative caches, respectively. In our

TABLE II. ENERGY SPECIFICATIONS OF CACHE MODULES

Main-cache	Main-tag	Mini-cache	Mini-tag
75.64 pJ	23.64 pJ	18.44 pJ	2.75 pJ

TABLE III. SPECIFICATIONS OF SMALL-TAG

Energy per access	Area
1.17 pJ	27,315 μm^2

STC architecture, a flip-flop-based Small-tag is used. If the size of the Small-cache and its corresponding address space are 1KB and 4MB, respectively, a single way of the Small-tag can be implemented using a 12-bit by 32-word flip-flop array, a 32-to-1 multiplexer and a 12-bit comparator. The energy consumption and area of Small-tag in our STC architecture is presented in TABLE III. As one can see from TABLE II and III, the energy consumption of the Small-tag is less than half of that of the Mini-cache tag memory. The area of the Small-tag is comparable with the area of a 512-byte SRAM module designed with the same commercial 65nm technology, which is almost twice of the area of the Mini-tag.

Four benchmark programs are used in our experiment; MPEG-2, JPEG, ADPCM coder, and FFT. The GNU C compiler and debugger for Toshiba MeP architecture are used for generating address traces. The length of the trace for each benchmark program is ten million instructions after skipping the initial one million instructions. TABLE IV shows the code size and the active code size for each benchmark program. The L0-cache, SPM architecture, HPC architecture and our STC architecture are evaluated. A trace-driven cache simulator is used for the evaluations.

B. Simulation Results

Figure 8 shows the energy consumption and performance results of the STC architecture and previous approaches presented in section II. For the SPM architecture, more frequently accessed functions are placed in the SPM. For the HPC and the STC architectures, placement of functions into the two cache regions is performed so that the total energy consumption of the memory subsystem is minimized. For ADPCM coder, the SPM architecture is the best of all because 55% of active code can be placed in the SPM for the application. However, this does not always happen. For MPEG-2 and JPEG, our approach outperforms all of the previous techniques. In JPEG, our STC architecture reduces the energy consumption of memory subsystems by 13% without performance degradation compared to the best result obtained by the previous approaches.

VI. CONCLUSIONS

A Single cycle accessible Two-level Cache (STC) architecture is proposed in this paper. Our future work will be devoted to come up with an algorithm for finding the best Small-cache configuration and code placement for the two cache regions simultaneously, which minimizes the energy consumption of memory subsystems.

TABLE IV. CODE SIZE AND ACTIVE CODE SIZE OF BENCHMARKS

Benchmark	Total code size	Active code size
MPEG-2	123.6 KB	16.4 KB
JPEG	98.8 KB	5.1 KB
ADPCM coder	31.0 KB	1.8 KB
FFT	7.3 KB	5.1 KB

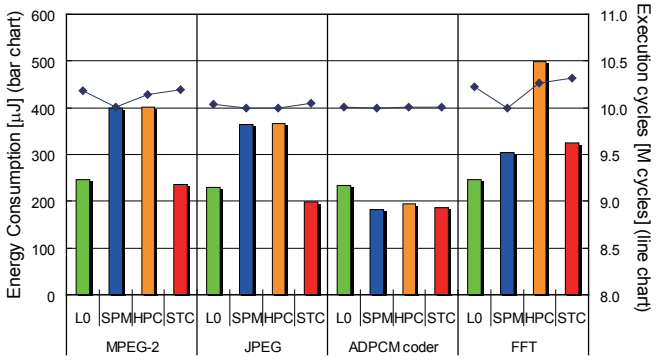

Figure 8. Energy consumption and execution cycles for each architecture.

ACKNOWLEDGMENT

The VLSI chip in this study supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with STARC, e-Shuttle, Inc., and Fujitsu Ltd Synopsys and Cadence Design Systems. This work is also supported by Toshiba, CREST ULP program of JST and JSPS Grant-in-Aid for Young Scientists (B) (20700049).

REFERENCES

[1] K. Ghose and M. B. Kamble. Analytical energy dissipation models for low power caches. In *Proc. of International Symposium on Low Power Electronics and Design*, pages 143–148, August 1997.

[2] A. González, C. Aliagas, and M. Valero. A data cache with multiple caching strategies tuned to different types of locality. In *Proc. of International Conference on Supercomputing*, pages 338–347, July 1995.

[3] J. Montanaro et al. A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE Journal of Solid-State Circuits*, 31(11):1703–1714, November 1996.

[4] R. Panwar and D. Rennels. Reducing the frequency of tag compares for low power i-cache design. In *Proc. of International Symposium on Low Power Electronics and Design*, pages 57–62, August 1995.

[5] S. Segars. Low-power design techniques for microprocessor. *International Solid-State Circuits Conference Tutorial*, February 2001.

[6] A. Shrivastava, I. Issenin, and N. Dutt. Compilation techniques for energy reduction in horizontally partitioned cache architectures. In *Proc. of International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 90–96, September 2005.

[7] S. Steinke, L. Wehmeyer, B.-S. Lee, and P. Marwedel. Assigning program and data objects to scratchpad for energy reduction. In *Proc. of Design, Automation and Test in Europe*, pages 409–415, March 2002.