Dynamic Management Technique to Mitigate Performance Degradation for Low-Leakage Caches

Komiya, Reiko
JSPS Research Fellow DC | Department of Informatics, Kyushu University | Department of Informatics, Kyushu University

Inoue, Koji
Department of Informatics, Kyushu University

Murakami, Kazuaki
Department of Informatics, Kyushu University

出版情報：Proceedings of Cool Chips X, pp.173-184, 2007-04-19
バージョン：
権利関係：

Dynamic Management Technique to Mitigate Performance Degradation for Low-Leakage Caches

Reiko Komiya † ‡, Koji Inoue †, Kazuaki Murakami †

† Department of Informatics, Kyushu University, ‡ JSPS Research Fellow DC
744 Motooka Nishi-ku, Fukuoka-shi, Fukuoka, 819-0395 Japan
Phone: +81-92-802-3794, Fax: +81-92-802-3786

Abstract

A number of techniques to reduce the cache leakage energy have so far been proposed. However, the low-leakage caches cause performance degradation. In order to solve this performance issue, we propose a new cache management technique. In this approach, a small number of cache lines which are frequently accessed in sleep mode are forced to stay in always-active mode. In this mode, transiting to the sleep mode is prohibited. In our evaluation, this approach can eliminate almost all performance overhead compared to conventional low-leakage cache without affecting the energy efficiency.
Keywords: High performance, Low leakage, and Cache

1. Introduction

Reducing the leakage in on-chip caches is essential, because of a tendency that a significant portion of transistor budget in microprocessors is invested to on-chip memories. To solve the leakage issue, a number of researchers have proposed efficient leakage reduction techniques. These techniques support two operation modes, a conventional high-leakage mode (or **active mode**) and an optimized low-leakage mode (or **sleep mode**). We call a cache line in sleep mode a **sleep-line**. Similarly, a line in active mode is referred to as an **active-line**. Gated-Vdd [3] is a representative implementation method of sleep-line. This method gates the supply voltage to sleep-line, and loses the data in the line. If the line which lost data is referred to again, the number of misses increases, and reference to low level memory increases. We call the miss which causes performance degradation "**extra-miss**". In our evaluation, we found that in the worst case, degradation in processor performance of 32.1% has been observed [2]. To solve the performance problem caused by extra-miss, we propose a new cache management technique. An abstract of this technique has been presented in [2]. In this paper, we describe the proposal technique in detail. This technique brings performance improvement by 14.8% without affecting the energy efficiency in the best case.

2. High-Performance, Low-Leakage Cache: Always-Active Scheme

It's apparent that a small number of lines are responsible for the majority of the performance degradation [2]. For example, in *183.equake*, lines of only 7.7% cause extra-misses of 75.2%. Based on this observation, we propose the "**always-active scheme**." This scheme prevents a cache line changing to sleep mode if the line cause extra-misses frequently. As a result, the number of extra-miss decreases and performance is improved. We refer to the state that does not transit to sleep mode as "**always-active mode**." In addition, a cache line in always-active mode is called an "**always-active line**." We call the access which causes extra-miss in a conventional low-leakage cache "**extra-miss-candidate**". We define the frequency of extra-miss-candidate to each cache line as extra-miss-candidate density (***EMCD***). The *EMCD* of line *i* is defined as follows:

$$EMCD_i = N_{extra-miss-condidate(i)} / N_{extra-miss-condidate(avg)} \quad (1)$$

where $N_{extra\text{-}miss\text{-}candidate(i)}$ is the total number of extra-miss-candidate occurring at cache line *i*, and $N_{extra\text{-}miss\text{-}candidate(avg)}$ is the average number of extra-miss-candidate for all cache lines. If a cache line indicates a value of *EMCDi* exceeding a given threshold, the line is considered to be responsible for a high proportion of extra-misses. Therefore, that line is assigned to the always-active mode.

A cache design supporting always-active scheme is presented in Fig.1. In this figure, the cache is assumed to be 1-way with 1024 lines. The area inside the broken line is the circuit for a conventional low-leakage cache: cache decay [1]. In the cache decay strategy, accesses to each cache line are monitored. If there are no accesses to a line during a given period, called the decay interval, the line transits to sleep mode (or decay flag is 1). The line which is referred to again is transit to active mode. If the decay interval has passed since the last access and tag hits, the access is extra-miss-candidate. Other circuit is

necessary for implementation of always-active scheme. In a conventional cache decay scheme, not only the data but the tag is destroyed, however only the data is destroyed in the always-active scheme. If the tag hits and the decay flag is 1, the local extra-miss-candidate counter and the global extra-miss-candidate counter are incremented. By checking the following expression during program execution, an always-active line can be determined dynamically:

$$N_{extra-miss-candidate(i)} > N_{extra-miss-candidate(avg)} \times Threshold \quad (2)$$

If Eq. (2) is satisfied, we set the always-active flag to 1, and the decay flag is masked. In other words, the line operates as an always-active line. And the extra-miss-candidate makes hit. On the other hand, if it is not satisfied, we reset the always-active flag, and the line operates based on decay flag same as cache decay.

3. Evaluation

We evaluate the energy-performance efficiency of the always-active scheme. The following caches are compared for the evaluation.

- **Decay**: a conventional cache decay.
- **AA2**: the cache lines whose *EMCDi* value is equal to or greater than 2 are assigned to the always-active mode. After experimentation, we found that a threshold of 2 represented a good balance between performance and energy.

The detail of the processor configuration is shown in Table 1. The leakage reduction technique is applied to an L1 data cache. We use the SimpleScalar simulator toolset, and six integer programs and four floating-point programs from the SPEC CPU 2000 benchmark set. In the cache simulation, the first one billion instructions are skipped to make execution stable, and the following 500 million instructions are used for the measurements.

Fig.2 shows the program execution time for the Decay and AA2. Each value is normalized to the normal cache which does not use any leakage reduction technique. In many benchmarks, the always-active scheme effectively compensates for the performance overhead caused by the cache decay. For example, in the case of *183.equake*, the performance reduction for AA2 is improved to 7.7% from 22.5% for Decay. On the other hand, Fig.3 indicates the breakdown of energy dissipation for the Decay, AA2 and a normal cache. LE_{L1}, DE_{L1}, and DE_{memory} are respectively, the leakage energy, the dynamic energy consumed in the data L1 cache, and the dynamic energy consumed with reference for main memory. The leakage energy and the dynamic energy which is consumed in additional circuits are included in LE_{L1} and DE_{L1} respectively. For *183.equake* and *175.vpr*, AA2 slightly reduces the energy compared to the Decay. In contrast, there is a little sacrifice in *179.art*, *188.ammp*, *176.gcc* and *181.mcf*. On average, AA2 achieve almost the same energy reduction rates as the Decay. Compared to the normal cache, Decay or AA2 is increased DE_{memory} and is decreased LE_{L1}. Therefore, the benefits of these schemes are dictated by the trade-off between the DE_{memory} reduction and the increase in LE_{L1}. Always-active scheme aggressively reduces DE_{memory}. Therefore, it improves performance and can reduce energy by performing suitable line selection.

If the memory access latency increases, Decay negative affects processor performance, due to accesses to sleep line. In that case, our proposal technique will become increasingly essential for performance improvement.

4. Conclusions

In this paper, the implementation of an “always-active scheme” was proposed, in order to achieve high-performance, low-leakage caches. In this approach, cache lines which cause the performance degradation are forced to stay in active mode. The energy-performance efficiency of the proposed method was evaluated. As a result, it has been observed that the approach can eliminate almost all performance overhead compared to a conventional cache decay scheme.

References

[1] S.Kaxiras, Z.Hu, and M.Martonosi, “Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power,” *Proc. of the 28th Int. Symp. on Computer Architecture*, pp.240-251, June 2001.

[2] R.Komiya, K.Inoue, K.Murakami, “Performance Optimization for Low-Leakage Caches based on Sleep-Line Access Density,” *4th Workshop on Optimizations for DSP and Embedded Systems*, March 2006.

[3] M.Powell, S.H.Yang, B.Falsafi, K.Roy, and T.N.Vijaykumar, “Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories,” *Int. Symp. on Low Power Electronic and Design*, pp.90-95, July 2000.

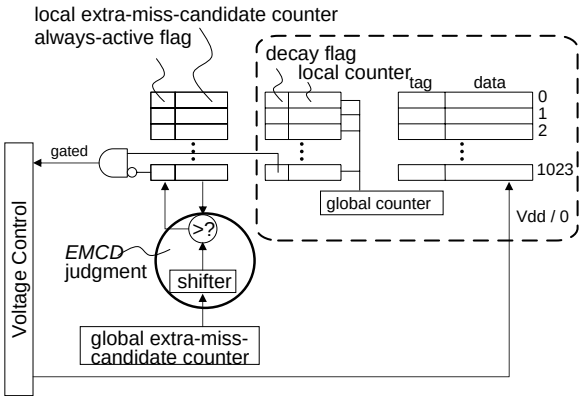

Fig. 1: Brock Diagram for the Always-Active Scheme

Table 1: Configuration of Processor

Instruction issue	In-Order
Instruction decode width	2 instructions / clock cycle
Instruction issue width	2 instructions / clock cycle
Cache memory	
L1 data L1 instruction	32KB (32B / entry, 32way, 1K entries) 32KB (32B / entry, 32way, 1K entries)
Hit latency	
L1 cache Main memory	1 clock cycle 64 clock cycles
Memory bandwidth	8B
Memory ports	1
ITLB, DTLB	
Entry Miss penalty	1M entries (4KB / entry, 32way, 32 entries / way) 30 clock cycles
Decay interval	8K clock cycles

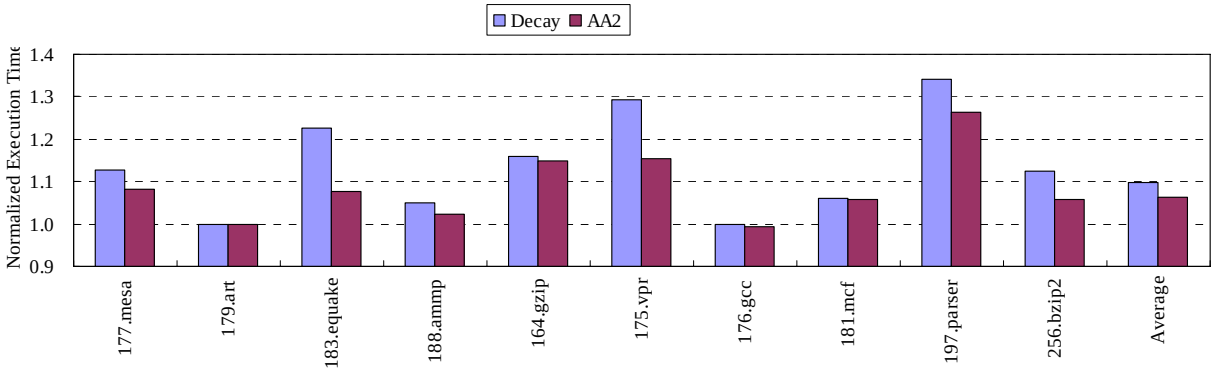

Fig.2: Normalized Execution Time

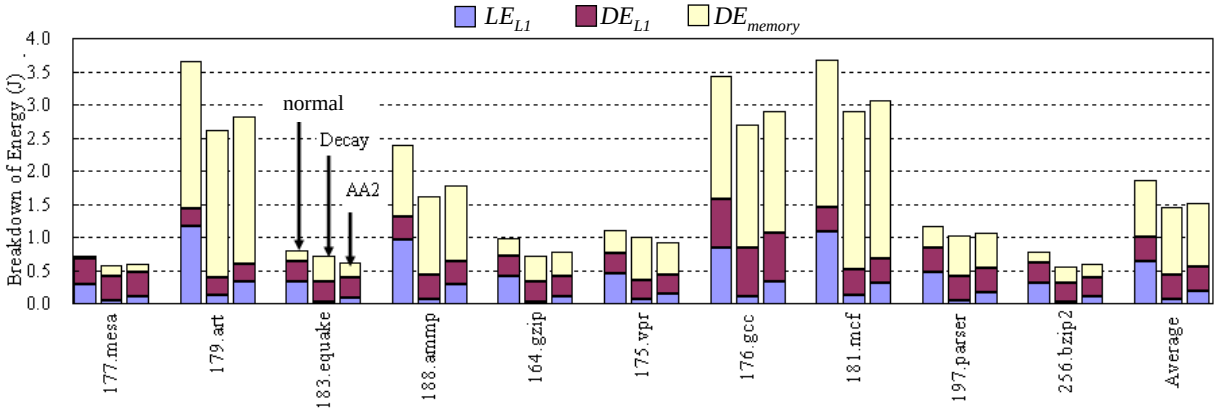

Fig. 3: Breakdown of Energy