A Leakage-Aware L2 Cache Management Technique for Producer-Consumer Sharing in Low-Power Chip Multiprocessors[*]

Hyunhee Kim and Jihong Kim

School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
+82-2-880-1861, {hh0726, jihong}@davinci.snu.ac.kr

Abstract As the process technology advances below 90nm, a leakage power management becomes a critical issue in realizing low-power chip-multiprocessors (CMPs). Especially, reducing the leakage power consumption of the L2 cache is important because most CMPs dedicate a large portion of their on-chip area for an L2 cache, making it a dominant leakage consumer. In this paper, we propose a novel leakage management technique for applications with producer-consumer sharing patterns. Exploiting particular access sequences observed in producer-consumer sharing patterns and the spatial locality of shared buffers, our technique enables a more aggressive turn-off of L2 cache blocks once the producer-consumer sharing pattern is detected during run time. Experimental results using a CMP simulator show the proposed technique reduces the energy consumption in the on-chip L2 caches and off-chip memory by 14.8% on average over the existing cache leakage energy management technique without a performance loss.

Keywords: leakage energy management, L2 cache, CMPs, producer-consumer sharing

Introduction Power dissipation becomes an important issue in designing modern CMPs. Especially, as the process technology advances below 90nm, the leakage power consumption becomes a dominant power consumer in the total power dissipation. Therefore, reducing leakage power consumption is a critical design goal for low-power CMPs. Since a large on-chip L2 cache is employed in most CMPs to hide a gap between the off-chip memory and processors, it dominates the leakage power consumption of the on-chip, thus making the leakage power management in a large L2 cache more important. Most existing cache leakage power management techniques (such as the *cache decay* technique [1]) turn off inactive cache blocks if they have not been accessed for predefined threshold cycles. Although the turn-off based leakage management scheme is simple and efficient, the existing leakage management scheme can be further improved by exploiting various run-time characteristics of target applications. In this paper, we focus on applications with producer-consumer sharing patterns. As reported in [8], producer-consumer data sharing is one of the most commonly used methods for multithreaded applications. Our technique is unique in that it took advantages of producer-consumer sharing in reducing the leakage energy consumption for CMP architectures. Fig. 1 shows an overall architecture of our target machine used in this paper. We assume that a target CMP is based on a MESI-like cache coherence protocol. Each processor is assumed to have a private L2 cache as a last level cache.

Leakage Energy Management Technique In our *leakage-aware L2 cache management technique for producer-consumer sharing* (LAPC) scheme, the main question is how to detect and maintain the consecutive addresses of shared buffers observed in producer-consumer sharing patterns. We describe an efficient detection technique that does not require much hardware by exploiting coherence transactions. We also present a history-based aggressive threshold setting technique for detected shared buffer L2 blocks.

For shared buffer detections, LAPC exploits a particular sequence of coherence transactions observed in producer-consumer sharing patterns. Fig. 2 shows a series of coherence transactions between a producer and a consumer, assuming a snoop based MESI protocol. When (1) the producer writes data in "A", one of the shared buffer addresses, (2) the coherence transaction "ReadX A" is generated to invalidate the copy of it in the consumer's cache if it exists. On the other hand, when (3) the consumer reads the data in "A", (4) "Read A" is generated for a read miss. In our technique, we detect this pattern in coherence transactions by using a global table and stream registers efficiently as shown in Fig. 3.

To detect the addresses which belong to the shared buffer, the global table maintains entries for an address and a set of bits representing the producer and consumer processor of the corresponding address. The FSM shown

[*] This work was supported in part by the Brain Korea 21 Project in 2008 and by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R0A-2007-000-20116-0). This work was supported by World Class University (WCU) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (R33-2008-000-10095-0). The ICT at Seoul National University provides research facilities for this study.

in Fig. 4 summarizes how each global table entry changes when “ReadX A” and “Read A” transactions appears on the bus. To manage the consecutive addresses of the detected shared buffer using a small hardware, the stream registers are also employed. (The stream registers were proposed in [7] for snoop filtering but we use them for storing the addresses of the shared buffer by exploiting the high spatial locality of them.) Each stream register, which represents a contiguous region R of the shared buffer, consists of the base address of R, mask, and producer and consumer bits. The mask whose zero indicates “don’t care” bit is used when deciding whether an address matches the base of the stream register or not. As shown in Fig. 5, when updating the stream register, a new register is allocated if an address from the global table matches none of the registers. Otherwise, it is added to the existing stream register updating the mask so that differing bit positions between the base and address can be “don’t-care” bit.

After a stream of the shared buffer is detected, LAPC applies a more aggressive turnoff threshold for shared buffer L2 blocks based on a weighted average of the past hit intervals (instead of a predefined turnoff threshold) which is updated whenever a read or a write hit to the shared buffer block occurs. When the cache blocks of the shared buffer in the producer’s cache are turned off, they are written to the consumer’s cache for future reuse. Although not discussed explicitly in this paper, LAPC can disable the shard buffer detection mechanism (e.g., by OS's control) when a target application is known to have no producer-consumer sharing. Note, however, our current estimate of the energy overhead of LAPC is negligible, taking less than 1% of the L2 cache energy, which was calculated by CACTI 4.1 [5].

Experiments To evaluate the proposed LAPC technique, we modified the CATS [2] simulator. Table I shows a machine configuration used for simulations. Table II summarizes the power parameters for the L2 cache and off-chip estimated from using [5] and [6]. Table III shows the benchmarks used in the experiment. All the programs in the benchmarks are from [3] and [4]. In a combination of four programs, each program is assumed to be a consumer for its left program and a producer for its right program while communicating with shared buffers.

Fig. 6 shows the normalized energy consumption of the L2 caches and off-chip memory where BASE indicates when no cache leakage management technique is used. The LAPC technique reduces the energy consumption in the private L2 caches and off-chip memory by 14.8% on average over the existing cache decay technique, DECAY, by turning off the cache blocks of the detected shared buffers aggressively than other cache blocks after the burst of the shared buffer accesses. Especially, LAPC can reduce the leakage energy consumption more compared to DECAY when the DECAY technique cannot turn off the cache blocks enough as shown in *bench2* and *bench4,* 16.6% and 21.3%, respectively. As shown in Fig. 7, the proposed scheme does not suffer any performance loss even though it turns off the cache blocks of the shared buffer earlier than other cache blocks. This indicates that our shared buffer detection technique works well although the stream registers detect the more addresses than the shared buffers used.

Conclusions Reducing leakage energy consumption of the on-chip L2 cache becomes an important issue in designing modern CMPs because most CMPs employ a large on-chip L2 cache for performance improvement. Although the existing cache leakage management technique can reduce the leakage energy consumption efficiently, we showed that exploiting the application knowledge, producer-consumer sharing in this case, can significantly improve the efficiency of leakage energy reductions. In this paper, we proposed LAPC that efficiently detects shared buffer accesses and aggressive turns off L2 blocks belonging to the detected shared buffers. Experimental results showed that the proposed technique can reduce the energy consumption in the L2 caches and off-chip memory by 14.8% on average over the existing technique without a performance loss.

[1] S. Kaxiras, Z. Hu, and M. Martonosi, “Cache decay: exploiting generational behavior to reduce cache leakage power,” Proc. of ISCA, pp. 240-251, 2001.

[2] D. Kim, S. Ha, and R. Gupta, “CATS: cycle accurate transaction-driven simulation with multiple processor simulators,” Proc. of DATE, pp. 749-754, 2007.

[3] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, “MediaBench: A tool for evaluation and synthesizing multimedia and communications systems,” Proc. of MICRO, pp.330-335, 1997.

[4] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” Proc. of WWC, pp. 3-14, 2001.

[5] T. David, T. Shyamkumar, and J. Norman, “Cacti 4.0: An integrated cache timing, power and area model,” http://www.hpl.hp.com/research/cacti/, 2006.

[6] Micron Technology, Inc., “Calculating memory system power for DDR,” 2005.

[7] V. Salapura, M. Blumrich, and A. Gara, “Improving the accuracy of snoop filtering using stream registers,” Proc. of MEDEA Workshop, pp.25-43, 2007.

[8] L. Cheng and H. B. Carter, and D. Dai, “An adaptive cache coherence protocol optimized for producer-consumer sharing,” Proc. of HPCA, pp.328-339, 2007.

Table I
Machine configurations used for simulations

Processor	4 Processors, in-order
L1 I/D-Cache	32 KB, 1-way, 32 B block, 1-cycle latency
L2 Private Cache	256 KB, 4-way, 128 B block, 6-cycle latency
Off-Chip Memory	300-cycle access latency
Global Table	256 entries, 26 bits/entry
Stream Registers	8 entries, 68 bits/entry

Table II
L2 cache and memory energy parameters

CMOS technology		70nm
Private L2 Cache	Dynamic read	0.11 nJ
	Dynamic write	0.01 nJ
	Leakage power	1802 mW
Off-Chip Memory	Dynamic read/write	2 nJ
	Standby Power	50 mW

Table III
Four benchmark combinations used in experiments

bench1	lu, fft, g721, blowfish
bench2	fft, mmul, lu, qsort
bench3	adpcm, mmult, fft_inv, qsort
bench4	lu, mmul, g721, qsort

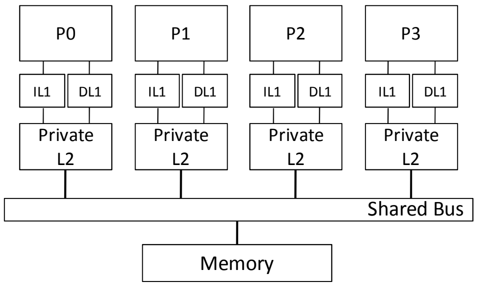

Fig. 1 Overview of target CMPs

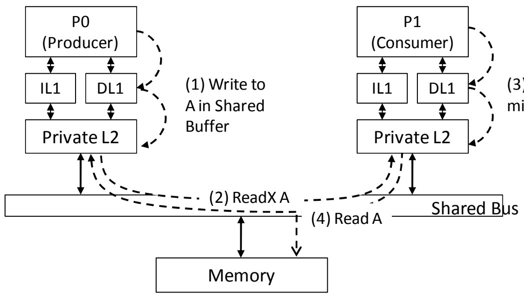

Fig. 2 Transactions between a producer (P0) and a consumer (P1)

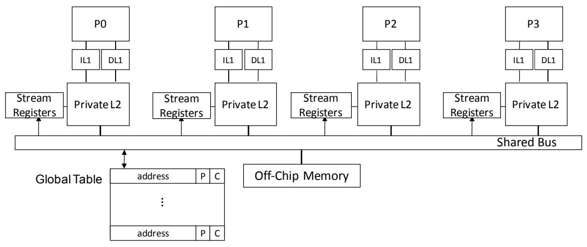

Fig. 3 Overall architecture of LAPC

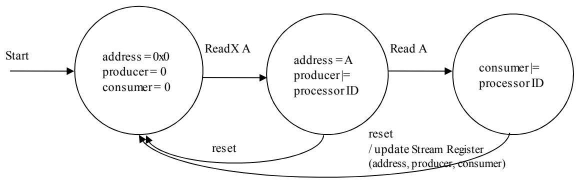

Fig. 4 FSM of a Global Table entry

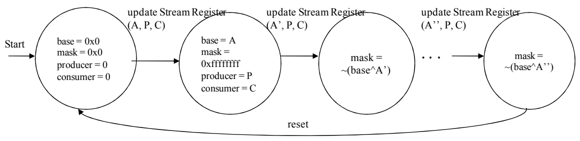

Fig. 5 FSM of a Stream Register

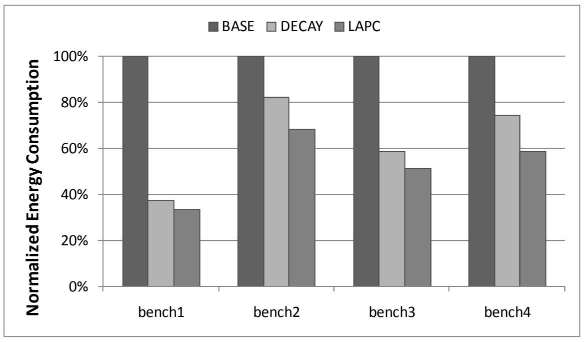

Fig. 6 Normalized energy consumption for the benchmarks

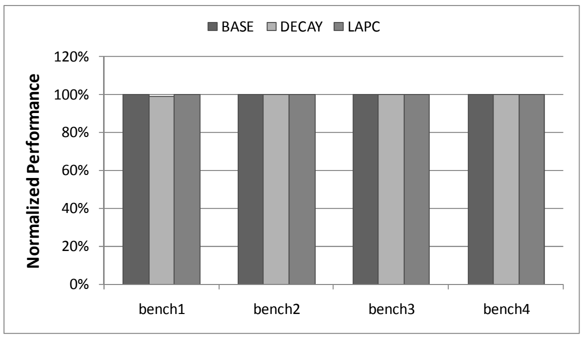

Fig. 7 Normalized performance for the benchmarks