Replication-Aware Leakage Management in Chip Multiprocessors with Private L2 Caches

Hyunhee Kim
School of CSE
Seoul National University
Seoul, Korea
hh0726@davinci.snu.ac.kr

Jung Ho Ahn
Department of ICS
Seoul National University
Seoul, Korea
gajh@snu.ac.kr

Jihong Kim
School of CSE
Seoul National University
Seoul, Korea
jihong@davinci.snu.ac.kr

ABSTRACT
Power dissipation has become a critical issue in modern chip multiprocessors (CMPs). Managing the leakage power of their L2 caches is particularly important in realizing low-power CMPs because most CMPs employ large L2 caches to hide the performance gap between processors and an off-chip memory while leakage power becomes a major portion in the total power dissipation of CMPs as process technology advances below 90 nm. We propose a replication-aware leakage management technique that selectively turns off a replicated block in a private L2 cache for leakage power reduction. Once a cache line is turned off, the data is lost, but its tag maintains the coherence state. The cost of an extra cache miss due to the turned-off replication is limited since the data of the cache line exists in another on-chip cache. Furthermore, the replicated block incurs no overhead if it is invalidated by other processors in order to maintain cache coherence. Our proposed technique can be implemented by slightly modifying the MESI protocol with a new turned-off shared coherence state. This state indicates that the corresponding block is shared by other caches but turned off. Experiments on a 4 processor CMP with private L2 caches show that the proposed technique reduces the energy consumption of the L2 caches and main memory by 20.0% on average without introducing significant performance loss over the existing cache leakage management technique.

Categories and Subject Descriptors
B.3.2 [**Memory Structures**]: Design Styles-Cache memories

General Terms
Design

Keywords
Chip Multiprocessors, L2 caches, Leakage power management

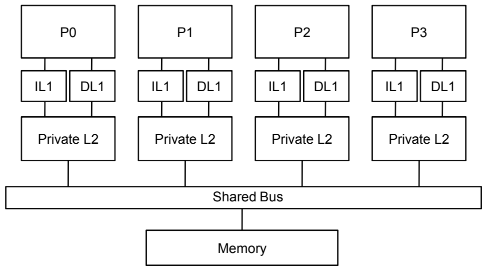

Figure 1: Target architecture of CMPs.

1. INTRODUCTION
Power dissipation has become one of the most important issues in designing modern microprocessors such as chip multiprocessors (CMPs). International Technology Roadmap for Semiconductor (ITRS) predicts that the leakage power consumption may constitute more than 50% of the overall power dissipation as the process technology drops below 65 nm [1], which means that the leakage power management should be a critical design goal for low-power CMPs. Since the on-chip L2 cache memories often determine the performance of CMPs, a large portion of their on-chip area is dedicated to them, which thus becomes a major power contributor. In this regard, reducing the leakage power consumption as well as improving the performance of the L2 caches for CMPs is a critical design issue.

Considerable body of research has proposed to reduce the leakage power consumption of caches [14, 9, 7, 3, 8, 12]. In these techniques, the leakage power consumption can be saved by gating off a SRAM cell as proposed in [14] to turn off inactive cells. Particularly, a cache decay technique [9] selectively turns off cache blocks that have not been accessed for threshold cycles as predicting them not to be accessed in the future. In this way, if the prediction is correct, the leakage power consumption can be saved during the turned-off period without a performance loss. However, it causes extra misses when the turned-off cache blocks are requested again since the data are not preserved. To overcome these drawbacks of the cache decay technique, the drowsy cache [7] supplies the minimum power to preserve data.

The cache decay technique is developed for L2 caches to save its large leakage power consumption for single processor systems [3]. It proposes a smart predictor to decide an adaptive threshold value for each cache block by monitoring the access interval between hits. For multiprocessor systems, [8, 12] are proposed. Virtual Exclusion [8] considers one of the multiprocessor characteristics, Multi-Level Inclusion. It reduces the leakage power consumption by turning off the repetitive cache blocks in the L2 caches when L1 caches have the same cache blocks, but requires that the sizes of the L1 and L2 cache blocks be the same. Monchiero et al. [12] also propose to reduce the leakage power of the L2 caches in CMPs by minimizing the possibility that dirty lines become turned off in order to improve the performance of decay.

In multiprocessor systems, several copies of the same memory block can coexist in more than one cache when multiple processors share the same memory block. We call such cache blocks *replications*. Although the existing leakage management techniques for CMPs are effective, they do not exploit the characteristics of these replications in CMPs. In terms of the performance improvement, an increasing amount of research also has proposed the techniques to selectively replicate cache blocks in the private L2 cache organization of CMPs [5, 4] to achieve a balance between capacity and latency. Cooperative Caching [5] proposes to replicate cache blocks with a given probability varying from 0% to 100%. ASR [4] also controls the replications to improve its performance by dynamically monitoring the benefits and costs of the replications. On the other hand, [16, 17] introduce self-invalidation in order to reduce cache coherence overhead caused by the replicated cache blocks. However, these techniques do not consider the energy consumption of the L2 caches, which is an important factor in designing low-power CMPs.

In this paper, we propose a replication-aware leakage management technique (RALM) which focuses on private L2 caches in CMPs. We assume a CMP with private L2 caches as shown in Figure 1. This organization has the lower access latency by replicating data close to the requesting processor but it reduces the on-chip capacity and causes more off-chip misses than the shared L2 cache organization. However, the private L2 cache organization has more benefits from the power perspective [5]: 1) a private L2 cache can be used as a unit for resource management to save energy when its processor is idle; 2) it can keep low set-associativity that consumes lower power.

Based on this power-efficient target architecture, the proposed technique selectively turns off a replicated block by exploiting its sharing characteristics. For example, a replicated block is often not likely to be accessed after another cache replicates it because it would be invalidated when its copy is updated. This sharing pattern allows us to reduce the leakage power without any performance loss by turning off the replicated block immediately after another cache replicates it. Furthermore, since the replicated block has its copy in another on-chip cache, turning it off has the benefit that the cost of an extra miss requires only an access to another on-chip private L2 cache rather than the off-chip access. The main advantage of the proposed RALM technique is that it can be implemented by slightly modifying the existing MESI cache coherence protocol without increasing implementation cost. Unlike the original MESI cache coherence protocol, our modified cache coherence protocol has one new state, Turned-Off-Shared (TOS), in which the cache block is shared by other processors but turned off.

Experimental results show that the proposed technique reduces the energy consumption by 20.0% on average over the existing leakage management technique while achieving a similar performance gain over a private L2 cache organization. The rest of this paper is organized as follows. In the next section, we introduce the motivation of our approach. Then, we describe the proposed replication-aware leakage management technique in Section 3 and experimental results are discussed in Section 4. We conclude the paper with a summary in Section 5.

2. MOTIVATION

The leakage power consumption has become a major contributor to the total power dissipation as the process technology advances. Although previous research can efficiently manage the leakage power consumption of L2 caches, they can be improved by exploiting characteristics of the cache blocks in CMPs. Our proposed technique is based on a simple observation that many of the replications can be turned off to save the leakage power consumption without negatively affecting the overall system performance. In this section, we describe how the proposed technique exploits replications to reduce the leakage power consumption by classifying L2 cache blocks into three categories depending on their sharing types: exclusive blocks, read-write replications, and read-only replications.

We call a cache block *an exclusive block* if it is not shared by any processor during its lifetime (i.e., from its first miss to its replacement). For replications that are shared by other processors, we further classify them into two groups depending on whether they are modified or not by other processors. We call a replication *a read-write replication* if there is at least one *write* request to any of its replications (including itself) during its lifetime. On the other hand, we call a replication *a read-only replication* if there are only read requests to all of its replications during its lifetime.

Our proposed technique aims to selectively turn off replicated blocks. For replicated blocks, we take advantage of the following observations:

1. In the target architecture, on a cache miss, data can be loaded into its local L2 cache from another processor's L2 cache (if it has the requested data) that has a shorter access latency than the off-chip memory. This allows that replications can be turned off aggressively to save the leakage power without decreasing the overall performance if it is guaranteed that their replications exist on-chip. Although the turned-off replications are needed again, they can be brought from another processor's cache.

2. For read-write replications, many of them are invalidated by other processors after they are replicated and their copies in the replicating processors' caches are modified. Furthermore, we observe that, in many cases, these are not accessed again once other processors *replicate* them. This allows us to turn them off *before they are invalidated* without any performance loss.

3. For read-only replications, unlike the read-write replications, they can be accessed again after being replicated. If they are turned off aggressively, this might cause extra misses. Although these extra misses require only accesses to another processor's on-chip cache, the performance can be degraded significantly if the turned-off read-only replications are frequently accessed. In this context, it is necessary that these replications

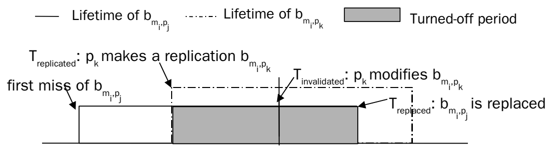
Lifetime of b_{m_i,p_j}
Lifetime of b_{m_i,p_k}
Turned-off period
$T_{replicated}$: p_k makes a replication b_{m_i,p_k}
first miss of b_{m_i,p_j}
$T_{invalidated}$: p_k modifies b_{m_i,p_k}
$T_{replaced}$: b_{m_i,p_j} is replaced

Figure 2: The lifetime of the replications, b_{m_i,p_j} and b_{m_i,p_k}, in the private L2 caches of the processors, p_j and p_k.

are selectively turned off if they might incur the performance drop.

Figure 7 shows the distribution of the L2 cache blocks with their sharing patterns. For `MPGdecoder` and `MPGencoder` of ALPBench [11], more than 80% of the cache blocks allocated during the program execution are replications. Among those replications, 52.9% and 72.0% of the blocks are classified as read-write replications while read-only replications account for 44.4% and 25.3% in `MPGdecoder` and `MPGencoder`, respectively. A large percentage of the replications are the main reason why the proposed technique is efficient. By turning off these replications efficiently, the proposed technique saves the leakage power consumption without decreasing the overall performance.

Furthermore, the proposed technique can be easily integrated into the existing MESI cache coherence protocol with small modifications. Since all of the actions required to turn on/off the replications are performed when the transactions in the original MESI cache coherence protocol occur, the proposed technique can be implemented with a small hardware overhead.

3. LEAKAGE MANAGEMENT BY TURNING OFF REPLICATIONS

3.1 Cache Decay Technique

In the proposed technique, the time-out based cache decay [9] technique is applied to all of the cache blocks except for replications. In order to keep track of the elapsed cycles from the last access, two levels of counters, global and local counters, are used. The global cycle counter sends a tick signal to the local counters every certain cycles, and the local counter of each cache block is incremented by one whenever it gets the tick. When the local counter reaches the time-out threshold, the corresponding cache block is turned off. In the experiment, we use 1 million cycles for the time-out threshold. We empirically observe that it is enough not to incur many extra misses in private L2 caches.

The proposed scheme is also based on the private L2 cache organization that uses the inclusion property [6], which is usually used in multi-level caches to efficiently implement a cache coherence in multiprocessor systems. To correctly maintain cache coherence and the inclusion property while employing the cache decay technique, we turn off only data portions of the cache blocks, keeping the tags and states of the blocks active. It allows only tags and states of the L2 caches to be responsible for maintaining the cache coherence as in a private L2 cache organization that does not employ the cache decay technique. Even though the proposed technique does not turn off the tags and states, it can

```
if (read miss for $m_i$ occurs in $p_k$) then
    if (another processor $p_j$ has a valid $b_{m_i,p_j}$) then
        if ($b_{m_i,p_j}$ is dirty) then
            write data back to the memory;
        end if
        turn off $b_{m_i,p_j}$;
        $C_{b_{m_i,p_j}} = 0$;
    end if
else if (read hit for $m_i$ occurs in $p_k$) then
    if ($b_{m_i,p_k}$ is shared and turned off) then
        if ($C_{b_{m_i,p_k}} < \tau$) then
            $C_{b_{m_i,p_k}}$ += 1;
        else
            turn on $b_{m_i,p_k}$;
        end if
    end if
else if (write hit for $m_i$ occurs in $p_k$) then
    if ($b_{m_i,p_k}$ is shared and turned off) then
        turn on $b_{m_i,p_k}$;
        invalidate other copies of $m_i$ on-chip;
    end if
end if
```

Figure 3: Algorithm of RALM.

significantly reduce leakage power because the energy consumption of the tags and states is an order of magnitude smaller than that of data blocks.

3.2 Algorithm of Replication-Aware Leakage Management (RALM)

We assume that m memory addresses $m_1, m_2, ..., m_m$ are referenced by n processors $p_1, p_2, ..., p_n$ during the execution in a parallel program. In this paper, a cache block for a memory address m_i, which is requested by a processor p_j, is denoted by b_{m_i,p_j}. When multiple processors, e.g., p_j and p_k, request the same memory address m_i, two cache blocks b_{m_i,p_j} and b_{m_i,p_k} are allocated in the L2 caches of p_j and p_k.

In order to describe the turned-off period of the replications when using the proposed RALM technique, Figure 2 shows the lifetime of two replications of a memory address m_i, b_{m_i,p_j} and b_{m_i,p_k}, which are allocated in the private L2 caches of the processors p_j and p_k. As for the read-write replications, the lifetime of b_{m_i,p_j} is overlapped by that of b_{m_i,p_k} during the time when b_{m_i,p_k} is replicated and b_{m_i,p_j} is replaced. If p_k modifies its copy b_{m_i,p_k}, b_{m_i,p_j} is invalidated before it is replaced. If b_{m_i,p_j} is not accessed during the period $T_{invalidated} - T_{replicated}$, it can be turned off immediately after being replicated without any performance loss. In the same way, the read-only replications can also be turned off during the period $T_{replaced} - T_{replicated}$. As explained above, however, these read-only replications can be accessed during the turned-off period. If many of them are frequently accessed during this period, the performance may decrease. Our proposed technique also considers this performance degradation and selectively turns off the replications in order to avoid it.

Figure 3 shows the algorithm of the proposed RALM technique, which includes operations that should be performed when *read miss*, *read hit*, and *write hit* occur in a processor p_k. On the cases that are not shown here, the cache operates in the way the same as the original private L2 cache that employs a MESI cache coherence protocol. When p_k

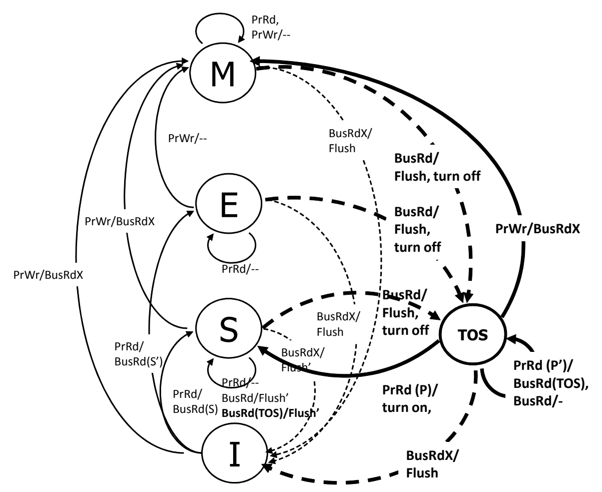

Figure 4: Modified cache coherence protocol: Bold dashed/solid lines represent additional transitions for RALM.

reads the memory block m_i but the read miss occurs in its L2 cache, the memory block is brought from the L2 cache of another processor or the off-chip memory. In this context, if another processor p_j (j $\neq$ k) already has a valid cache block b_{m_i,p_j}, it flushes its data to p_k while turning off b_{m_i,p_j}. It should be noted that if b_{m_i,p_j} is dirty, the data is written back to the off-chip memory before the cache block is turned off.

In order to avoid the performance degradation explained above, we employ an access counter for each cache block, e.g., $C_{b_{m_i,p_j}}$ for b_{m_i,p_j}, which keeps track of a number of accesses that occur after its cache block is turned off. This $C_{b_{m_i,p_j}}$ is reset to zero when the cache block is turned off. When p_k reads m_i but it has b_{m_i,p_k} that is turned off, the proposed technique only increments $C_{b_{m_i,p_k}}$ by one instead of turning on the cache block while fetching the data from another processor's cache (or from the off-chip memory if it does not exist) if its $C_{b_{m_i,p_k}}$ is less than a threshold τ. However, if $C_{b_{m_i,p_k}}$ reaches τ, the proposed technique turns on the corresponding cache block to avoid the performance degradation. On the other hand, when the replication is modified after being turned off, it should be turned on to keep the new data as its $C_{b_{m_i,p_k}}$ is reset to zero.

3.3 Modified Cache Coherence Protocol

In order to efficiently turn off replications without complex hardware implementation, we integrate the proposed technique into the original cache coherence protocol by slightly modifying it. Figure 4 shows a state transition diagram for the modified MESI cache protocol for RALM.

The original MESI protocol consists of four states: modified (M), exclusive (E), shared (S), and invalid (I). In the diagram, the notation "A/B" indicates that when the controller observes the transaction "A" from a processor or bus, it generates the bus transaction or action "B" while changing the state. "-" means that no action occurs. For both of the original and modified protocols, the solid arcs represent transitions due to local processor transactions while the dashed arcs represent transitions due to bus transactions. In the original MESI, when a cache block is first read by a local processor, *PrRd* in the diagram, it enters the S state if the copy of it exists in another cache. In this case, it also generates the *BusRd* transaction to make the other copies enter the S state. If its copy does not exist, it enters the E state. When the cache block in the E state is written by a local processor, *PrWr*, it can transition to the M state without generating a bus transaction because no other cache has a copy while writing to the cache block in the S state generates the *BusRdX* transaction on the bus to invalidate the copies in other caches.

On the other hand, in the modified version, the Turned-Off-Shared (TOS) state is added to the original one. In this state, only the data block is turned off while its tag is kept turned on but treated the same as in the S state. The only difference from the S state is that when the cache access occurs, the data is brought from the other caches as the miss occurs. This additional state is needed in order to maintain the access counters for each L2 cache block. In the original protocol, the cache block enters the S state from M, E, or S when the *BusRd* transaction is presented on the bus while supplying the data to the requesting cache. However, in the modified version, when the cache block at one of those states receives the *BusRd* transaction from the bus, its state is changed to the TOS state instead of the S state while the data block of it is turned off. In this case, the data is flushed to the requesting cache in the same way as the original one.

The cache block in the TOS state does not change its state when the *PrRd* transaction from the local processor is presented, which means that it remains continuously turned off but only its access counter is incremented by one. The requested data is fetched from another processor's cache by generating a *BusRd* transaction along with a TOS signal, which does not turn off the cache block. The TOS block is turned on only when either of these two following transactions occurs. First, when the *PrWr* transaction from the local processor occurs, the cache block is turned on while changing its state to M. Second, when the *PrRd* transaction occurs and the access counter of the TOS block reaches τ, the data block is turned on while changing its state to S because it is the frequently accessed block. Whether the access counter reaches τ or not is indicated as "P" in the diagram.

Figure 5 shows the organization of RALM. The global counter sends a tick signal to the local counters (one per cache line) as in the cache decay technique. However, in RALM, Power Mode Control (PMC) is added to turn off the cache block based on the local counter signal and the state transition. When the local counter reaches the maximum value or the state of the cache block is changed to TOS or I, the cache block is turned off. The additional circuit is not on the critical path because the state transition time is the same as in the original cache protocol and the comparison between the access counter and τ can be processed when the tag matching occurs.

4. EXPERIMENT

4.1 Simulation Environment

To evaluate our technique, we modified the CATS [10] multiprocessor simulator to use a snoop based MESI protocol for the cache coherency that supports cache-to-cache transfer of the cache block among private L2 caches. We use in-order 4 processors, a current trend in the multiprocessor architecture that employs simple multiple processors

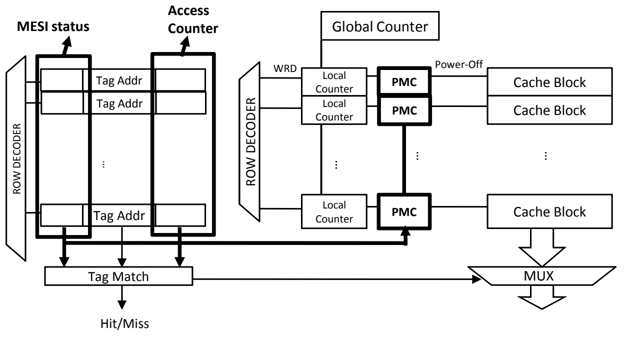

Figure 5: RALM organization: Bold solid line represents the additional structures and PMC stands for Power Mode Control.

Table 1: Architectural parameters.

Processor	4 Processors, in-order
L1 I/D-Cache	32 KB, 2-way 32 B block, 1 cycle latency
L2 Private Cache	512 KB, 4-way, 128 B block 6 cycle latency
Shared Bus	4 bytes bus width
Off-Chip Memory	300 cycle access latency

for low-power consumption. The energy parameters are obtained from CACTI 6.0 [13] using the 70 nm technology and the off-chip memory energy estimation [2]. We evaluated our scheme with 5 benchmarks, `MPGdecoder` and `MPGencoder` from ALPBench [11] and `VOLREND`, `RAYTRACE`, and `CHOLESKY` in SPLASH2 [15].

4.2 Results

We evaluate the proposed technique, RALM, by comparing it to BASELINE and DECAY. BASELINE indicates the baseline private L2 cache technique that does not employ any leakage management technique but the low power mode is applied to all of the cache blocks during the execution. Other techniques evaluated in the experiment also employ the low power mode when the cache blocks are active. Even though it increases the hit time, it can save the leakage power consumption. In the DECAY technique, the time-out based leakage management technique proposed in [9] is applied to all inactive cache blocks in each private L2 cache. We also evaluate RALM_NSEL that does not control the TOS blocks, as well as the proposed RALM technique, in order to see how the performance decreases when the replications are always turned off. In the experimental results, we only show the results that use 16 for τ as a threshold value in RALM. We evaluated other values for τ and found that the largest energy reduction is achieved when 16 is used with only a small hardware overhead in our configuration. In addition, for the DECAY and proposed RALM and RALM_NSEL techniques, the cache blocks are also turned off when they enter the I state.

Figure 6 shows the execution time and energy consumption of the L2 cache and the memory for each technique normalized to BASELINE. The DECAY technique shows the negligible performance degradation because we use the time-out threshold that is long enough not to incur many ex-

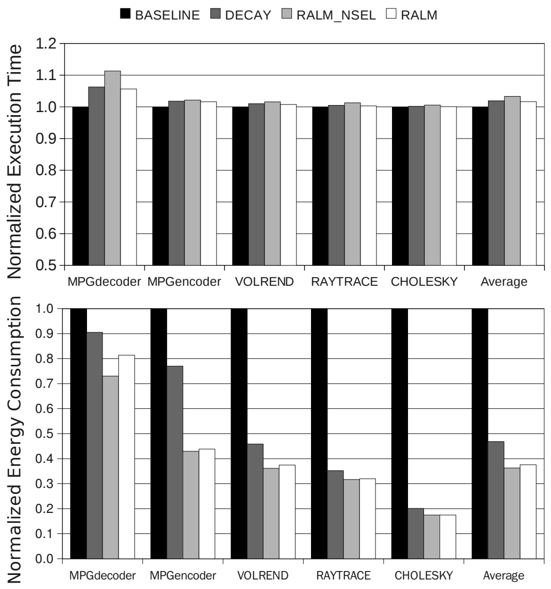

Figure 6: Normalized execution time and energy consumption.

tra misses. In RALM_NSEL, the performance degradation is less than 2% for most of the benchmarks except for `MPGdecoder`. For `MPGdecoder`, the performance degrades by up to 11.3% because many read-only replications of this benchmark, especially for the instructions of IDCT, are frequently accessed and they affect the performance consequently. On the other hand, in RALM, the performance degradation becomes similar to the DECAY technique because it can prohibit the replications that are frequently accessed from entering the TOS state.

We considered both of the dynamic and leakage power consumption of the L2 cache and the memory. For the dynamic energy consumption, it includes the dynamic energy consumption of the extra L2 cache and memory accesses caused by turning off the replications too early in the proposed technique as well as turning off the cache blocks in the cache decay technique. We also take into account the dynamic and leakage power consumption overhead of the additional counters for the proposed technique. In addition to the 10-bit global counter and the 10-bit local counters used in the cache decay technique, the proposed technique requires the 4-bit access counter for controlling the TOS block and 1 additional bit per cache block for the new state in the modified MESI protocol, which increases by 1% of the energy consumption and area of the total cache in comparison with BASELINE. The extra logic for the Gated-Vdd technique [14] is also considered, which becomes less than 3% of the total cache leakage.

As can be seen in Figure 6, RALM_NSEL reduces the energy consumption by 63.7% and 22.6% on average over BASELINE and DECAY, respectively. In particular, for `MPGdecoder` and `MPGencoder`, RALM_NSEL can reduce the energy consumption by up to 19.3% and 44.2% over DECAY while the SPLASH2 benchmarks show smaller reduction. In the proposed technique, the amount of reduction in the leakage power consumption depends on the percentage of replications. The more replications exist, the more leakage power can be reduced. Figure 7 shows how many replications are

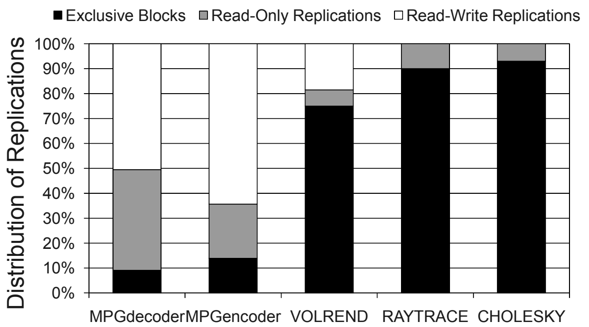

Figure 7: Distribution of replications.

allocated during the execution of each benchmark depending on their sharing patterns. For the benchmarks such as `MPGdecoder` and `MPGencoder` that have a larger number of replications, the proposed technique can reduce the energy consumption significantly. On the other hand, since the scientific benchmarks such as `CHOLESKY` in SPLASH2 do not have many replications, it has less opportunity to turn off the cache blocks.

For `MPGdecoder`, the performance degradation shown in Figure 6 is caused by a large number of read-only replications because they should be read from the other on-chip cache whenever a read miss occurs if they are turned off while the read-write replications do not cause the significant performance degradation. However, by controlling the number of TOS blocks, RALM reduces the energy consumption by 62.4% and 20.0% on average over BASELINE and DECAY, respectively. Although the energy reduction is smaller than that in RALM_NSEL because RALM selectively turns off the replications, it can keep its performance similar to that of DECAY technique.

5. CONCLUSIONS

We proposed a replication-aware leakage management technique that is based on a power-efficient private L2 cache organization for CMPs. The proposed technique turns off the replications immediately after the other on-chip cache makes the copy of it. Turning off the replications reduces the leakage energy consumption without significant performance loss because the cost of an extra miss only requires the on-chip access that is much faster than the off-chip access and the many replications are invalidated after making their copies in other caches. To turn off the replication efficiently without much hardware overhead, we slightly modified an original MESI cache coherence protocol. The experimental results show that RALM reduces the energy consumption by 62.4% and 20.0% compared to the BASELINE and DECAY technique on average without significant performance degradation.

6. ACKNOWLEDGEMENTS

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (No. R0A-2007-000-20116-0) and World Class University (WCU) program through KOSEF funded by the Ministry of Education, Science and Technology (No. R33-2008-000-10095-0). This work was also supported by the Brain Korea 21 Project in 2010 and Research Settlement Fund for the new faculty of SNU. The ICT at Seoul National University and IDEC provided research facilities for this study.

7. REFERENCES

[1] ITRS (International Technology Roadmap for Semiconductor). *http://public.itrs.net.*

[2] Calculating Memory System Power for DDR. *Micron Technology Inc.*, 2005.

[3] J. Abella, A. González, X. Vera, and M. O'Boyle. IATAC: A Smart Predictor to Turn-off L2 Cache Lines. In *TACO*, 2(1):55-77, 2005.

[4] B. M. Beckmann, M. R. Marty, and D. A. Wood. ASR: Adaptive Selective Replication for CMP Caches. In *Proc. of Micro*, pages 443-454, 2006.

[5] J. Chang and G. S. Sohi. Cooperative Caching for Chip Multiprocessors. In *Proc. of ISCA*, pages 357-368, 2006.

[6] D. E. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach.* Morgan Kaufmann, 1999.

[7] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy Caches: Simple Techniques for Reducing Leakage Power. In *Proc. of ISCA*, pages 148-157, 2002.

[8] M. Ghosh and H. S. Lee. Virtual Exclusion: An Architectural Approach to Reducing Leakage Energy in Caches for Multiprocessor Systems. In *Proc. of ICPADS*, pages 1-8, 2007.

[9] S. Kaxiras, Z. Hu, and M. Martonosi. Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power. In *Proc. of ISCA*, pages 240-251, 2001.

[10] D. Kim, S. Ha, and R. Gupta. CATS: Cycle Accurate Transaction-driven Simulation with Multiple Processor Simulators. In *Proc. of DATE*, pages 749-754, 2007.

[11] M.-L. Li, R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes. ALPBench Benchmark Suite for Complex Multimedia Applications. In *Proc. of IISWC*, pages 34-45, 2005.

[12] M. Monchiero, R. Canal, and A. González. Using Coherence Information and Decay Techniques to Optimize L2 Cache Leakage in CMPs. In *Proc. of ICPP*, pages 1-8, 2009.

[13] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi. CACTI 6.0: A Tool to Model Large Caches. In *http://www.hpl.hp.com/research/cacti*, 2009.

[14] 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. In *Proc. of ISLPED*, pages 90-95, 2000.

[15] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proc. of ISCA*, pages 24-36, 1995.

[16] A. R. Lebeck and D. A. Wood. Dynamic Self-Invalidation: Reducing Coherence Overhead in Shared-Memory Multiprocessors. In *Proc. of ISCA*, pages 48-59, 1997.

[17] A.-C. Lai and B. Falsafi. Selective, Accurate, and Timely Self-Invalidation Using Last-Touch Prediction. In *Proc. of ISCA*, pages 139-148, 2000.