IEEK Transactions on Smart Processing and Computing

Energy Consumption Evaluation for Two-Level Cache with Non-Volatile Memory Targeting Mobile Processors

Shota Matsuno[1], Masashi Tawada[1], Masao Yanagisawa[1], Shinji Kimura[1], Tadahiko Sugibayashi[2], and Nozomu Togawa[1]

[1] Waseda University / Tokyo, Japan {shota.matsuno, togawa}@togawa.cs.waseda.ac.jp
[2] NEC Corporation / Ibaraki, Japan

* Corresponding Author: Shota Matsuno and Nozomu Togawa

Received May 13, 2013; Revised May 28, 2013; Accepted June 12, 2013; Published August 31, 2013

* *Extended from a Conference: Preliminary results of this paper were presented at the ITC-CSCC 2013. This present paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

***Abstract*:** A number of systems have several on-chip memories with cache memory being one of them. Conventional cache memory consists of SRAM but the ratio of static energy to the total energy of the memory architecture becomes larger as the leakage power of traditional SRAM increases. Spin-Torque Transfer RAM (STT-RAM), which is a variety of Non-Volatile Memory (NVM), has many advantages over SRAM, such as high density, low leakage power, and non-volatility, but it consumes too much writing energy. This study evaluated a wide range of energy consumptions of a two-level cache using NVM partially on a mobile processor. Through a number of experimental evaluations, it was confirmed that the use of NVM partially in the two-level cache effectively reduces energy consumption significantly.

***Keywords*:** Non-volatile memory, Cache, Energy consumption

1. Introduction

On-chip memories, such as cache or scratchpad memory, are used widely because they play important roles as an absorber in recent processor systems to mitigate a gap between the processor core and main memory. The most widely used on-chip memory is cache, and conventional cache consists of SRAM. SRAM has been the optimal memory for cache because it has high access speed, low access power and high endurance. On the other hand its leakage power is increasing as the technology node becomes smaller, which has become a serious concern in low-power systems.

Non-Volatile Memory (NVM) not only holds data without a power source it consumes very low leakage power. It has many advantages over SRAM. In particular, Spin-Torque Transfer RAM (STT-RAM), which is a variety of NVM, has high access speed, high density and high endurance. STT-RAM, however, has very low writing performance in terms of energy consumption. Although it is called NVM, its characteristics follow that of STT-RAM. To the best of the author's knowledge, there are no reports on NVM energy consumption evaluations using a wide range of applications with various cache configurations*. Such NVM energy consumption needs to be evaluated.

In this paper, two-level cache and single-core processor were assumed to be on the same chip. A wide range of energy consumption of a two-level cache was also evaluated using STT-RAM-based NVM partially on a mobile processor. The use of NVM partially in two-level cache effectively reduces the overall energy consumption.

The remainder of this paper is organized as follows: Section 2 describes the related works. Section 3 explains the target architecture and computation model to prepare an energy consumption evaluation. Section 4 shows the energy evaluation results running in a number of application programs and a wide variety of cache configurations. Finally, the last section reports the conclusions.

* In [8], NVM cache energy consumption was evaluated on a single application program, and in [9] it was evaluated on three cache configurations.

2. Related Works

This section summarizes several existing studies related to NVM and STT-RAM.

Early Write Termination (EWT) is a technique to reduce the writing energy of STT-RAM [15]. Most parts of the bits are written on the same value that is already stored in the cache. These writing processes are redundant and an EWT circuit can terminate the redundant bit writing processes at their early stages. EWT can reduce the writing energy and dynamic by up to 80% and 67%, respectively.

For storage-class memory, it is important to keep the data at room temperature for at least 10 years. On-chip memory, however, does not need to keep the data for some years, and keeping it for a few seconds is sufficient in many cases. In [11], a technique to reduce the writing current and relax the retention time was proposed.

SRAM has high reading and writing access speeds and low reading and writing energy, but has high leakage power and low density. On the other hand, NVM, such as STT-RAM and PRAM have high density and low leakage power, but has a low writing access speed and high writing energy. Hybrid on-chip memory is a way of improving those disadvantages using the different characteristics of each memory. Introducing a hybrid Scratch-Pad Memory (SPM) by combining SRAM and NVM [4] realizes a larger capacity, lower leakage power, and lower writing energy than pure SRAM-based SPM. The use of a hybrid cache consists of SRAM and PRAM [6], and it also has better performance than a pure SRAM-based cache.

3. Target Architecture

The target architecture was assumed to be a single-core processor with a two-level cache. As shown in Fig. 1, a two-level cache has a L1 instruction cache (IL1) and L1 data cache (DL1) as the first layer and L2 unified cache (UL2) as the second layer. The cache replacement policies were assumed to be LRU for IL1, LRU for DL1 and FIFO for UL2. Every level was also assumed to have a write-through function.

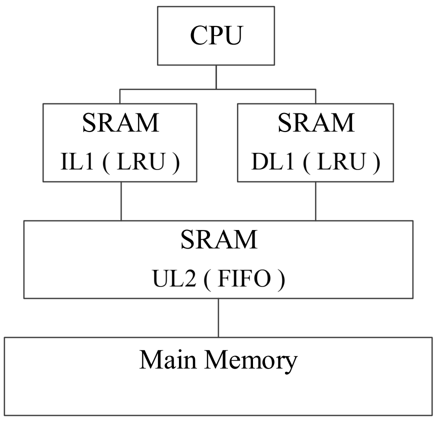


Fig. 1. SRAM-based cache architecture.

3.1 Using NVM for Cache

Cache is accessed quite frequently because it has a role as a working space. The energy consumed by reading and writing on cache affects the dynamic energy. Conventional cache consists of SRAM because SRAM has low access energy and high endurance. On the other hand, the leakage power of SRAM increases as the technology node becomes smaller, and the static energy of cache also increases. Recently, static energy has become too large to ignore in SRAM. The use of NVM instead of SRAM as a cache architecture is one suitable choice because it has very low leakage power.

STT-RAM, a form of NVM, is a promising option for next-generation cache. STT-RAM utilizes magnetic resistance to store data and spin-torque to write the logical value. STT-RAM has better performance than SRAM, such as fast reading and high endurance. Moreover, the leakage power of STT-RAM is quite small. On the other hand, STT-RAM consumes too much energy in a writing operation. For example, STT-RAM consumes 5 to 15 times more energy than SRAM [2, 11, 12, 15].

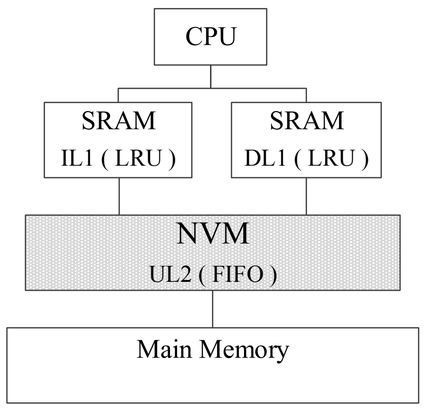
3.2 Two-Level Cache Architecture Using Non-Volatile Memory

Cache is accessed frequently as mentioned above, but frequency differs among the caches being used, such as IL1, DL1, and UL2. The upper cache is generally read or written more frequently than the lower cache and frequent cache access consumes dynamic energy. Because NVM consumes higher writing energy than SRAM, the dynamic energy consumption of NVM may become larger than that of SRAM. On the other hand, the static energy of NVM is exceedingly lower than that of SRAM, and then the use of NVM can improve the static energy consumption. The strategy to reduce the overall cache energy is to use a NVM-based cache to reduce the writing access.

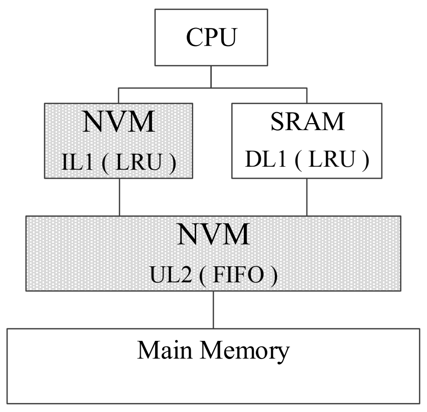
Data cache is being written frequently. If NVM is used as the data cache, its energy consumption might be increased because the writing energy is larger. The use of NVM as DL1 cache is not a suitable choice.

On the other hand, the instruction cache is generally used only for reading because the program codes are fixed at the run-time. Although the writing energy of NVM is larger than that of SRAM, its effects can be smaller when NVM is used as an instruction cache. Moreover, the overall cache energy can be reduced using NVM as an instruction cache because NVM has low leakage power. The use of NVM to IL1 cache instead of SRAM can reduce the energy consumption.

In the case of two-level cache, the size of the lower cache is generally larger than that of the upper one and the frequency of cache access to the lower cache is less than that of the upper one. Regardless of the access frequency, the leakage power becomes larger as the cache size becomes larger. Accordingly, the leakage power of the lower cache can be larger than that of the upper cache. Because NVM has very low leakage power, it should be possible to reduce leakage power significantly using NVM instead of SRAM to the lower cache. The use of NVM as



(a) NVM-based cache architecture (Type I): SRAM-based UL2 cache is replaced with NVM.



(b) NVM-based cache architecture (Type II): SRAM-based IL1 and UL2 caches are replaced with NVM.

Fig. 2. NVM-based cache architectures.

the UL2 cache instead of SRAM can reduce energy consumption.

Based on these discussions, three cache architectures were assumed, as shown in Figs. 1 and 2. Fig. 1 shows conventional *SRAM-based cache* architecture. SRAM-based UL2 cache is replaced with NVM-based UL2 cache, as shown in Fig. 2(a). A *NVM-based cache (Type I)* architecture is achieved because the frequency of UL2 cache access is less than that of IL1 or DL1 cache, and the leakage power of UL2 cache is larger than that of IL1 or DL1 cache. Fig. 2(b) illustrates the situation where SRAM-based IL1 and UL2 caches is replaced with NVM-based caches. A *NVM-based cache (Type II)* architecture is produced because a change in a program code basically would not occur when the program is running.

3.3 Cache Parameters

Cache can be characterized by its number of sets, block size and associativity, and can be written as (s, b, a), where s stands for the number of sets, b stands for the block size in bytes, and a stands for its associativity. The performance of cache, such as latency and energy consumption can be determined by (s, b, a).

Each of the three cache architectures, *SRAM-based cache*, *NVM-based cache (Type I)* and *NVM-based cache (Type II)*, has a two-level cache architecture consisting of three caches, IL1, DL1 and UL2. They can be written as c = ((s_i, b_i, a_i), (s_d, b_d, a_d), (s_2, b_2, a_2)), where (s_i, b_i, a_i) stands for the IL1 cache, (s_d, b_d, a_d) stands for the DL1 cache and (s_2, b_2, a_2) stands for the UL2 cache. The capacity of the two-level cache can be expressed as $C(c) = (s_i \times b_i \times a_i, s_d \times b_d \times a_d, s_2 \times b_2 \times a_2)$. Note that even if there are two caches whose capacities are the same, their cache performance cannot be the same because each of the cache parameters is different from each other.

3.4 Energy Computation Model

Consider the dynamic and static energy of every single cache that comprises the two-level cache when running a single application. The dynamic energy can be calculated by the product of the energy for each reading or writing memory access and the number of its accesses. The static energy depends on the leakage power and application run-time.

Multi-level cache can be considered a set of single cache. The energy consumption of multi-level cache is the sum of the energy of each single cache. The two-level cache consists of three caches, IL1, DL1, and UL2. The total energy E of the two-level cache is expressed by Eq. (1):

$$E = E_{IL1} + E_{DL1} + E_{UL2} \tag{1}$$

where E_{IL1} stands for the IL1 cache energy, E_{DL1} stands for the DL1 cache energy, and E_{UL2} stands for the UL2 cache energy.

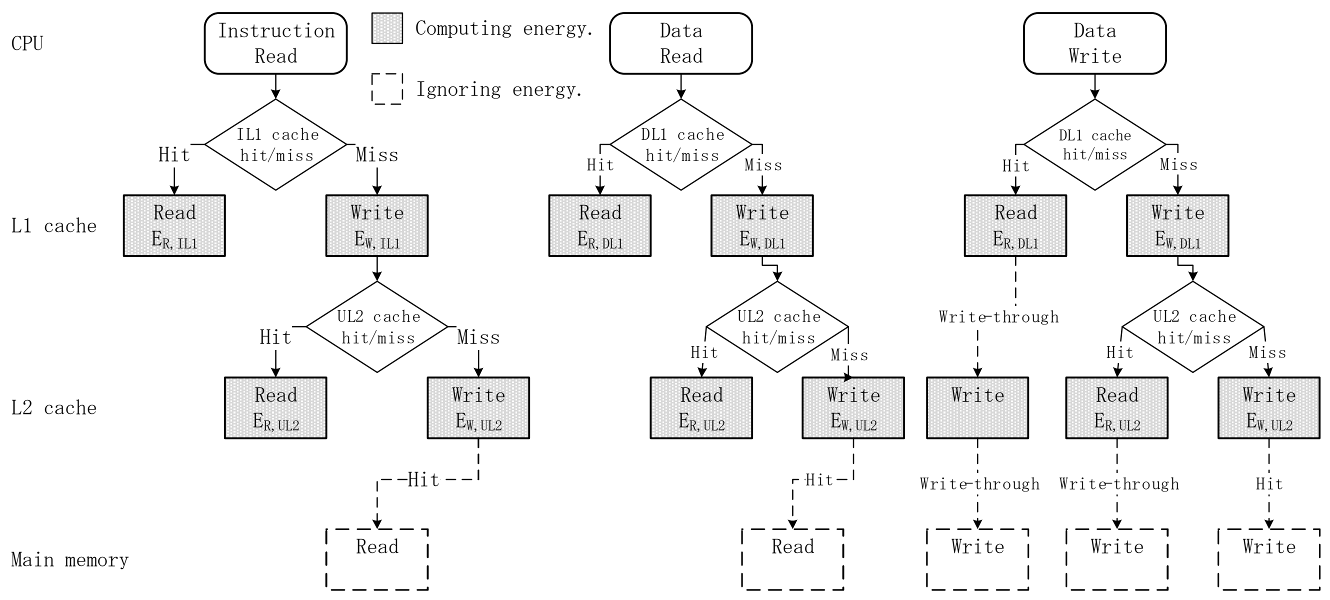
3.4.1 Energy computation of IL1 cache

E_{IL1} was computed using Eq. (2):

$$\begin{aligned} E_{IL1} = & E_{R,IL1} \times N_{I,H,IL1} \\ & + E_{W,IL1} \times N_{I,M,IL1} \\ & + P_{leak,IL1} \times T_{total} \end{aligned} \tag{2}$$

where $E_{R,IL1}$ stands for the reading access energy of the IL1 cache, $E_{W,IL1}$ stands for the writing access energy of the IL1 cache, $P_{leak,IL1}$ stands for the leakage power of the IL1 cache, $N_{I,H,IL1}$ stands for the number of cache hits on the IL1 cache, $N_{I,M,IL1}$ stands for the number of cache misses on the IL1 cache, and T_{total} stands for the total cache access time, which will be described in Section 3.4.4.

Fig. 3(a) shows the cache energy, which is composed of the instruction memory accesses and requires dynamic energy at the gray boxes in Fig. 3(a). In the case of the instruction accesses, a processor core accesses the IL1 cache at first. If a cache hit occurs on the IL1 cache, the reading energy of IL1 is needed to read the data from the



(a) A case of instruction access. (b) A case of data reading access. (c) A case of data writing access.

Fig. 3. Energy computation procedure of the two-level cache.

IL1 cache shown by the first term in Eq. (2). If we have a cache miss on the IL1 cache, the processor core accesses the UL2 cache and the writing energy of the IL1 cache for data replacement shown by the second term in Eq. (2) is needed. If a cache miss occurs on the UL2 cache, the processor core accesses the main memory and the writing energy of UL2 cache is needed for data replacement. The static energy of the IL1 cache is not dependent on the memory accesses but can be calculated using the product of the IL1 leakage power $P_{leak,IL1}$ and the total cache access time T_{total} shown by the third term in Eq. (2).

(a) Memory parameter computation

The parameters used in Eq. (2) can be calculated as follows. A single cache, which has (s, b, a) as the cache parameters, was assumed. The performance parameters ($E_{R,IL1}$, $E_{W,L1}$, $P_{leak,IL1}$) of the SRAM caches were calculated using CACTI [10]. The technology node used was 45 nm. The access time of the main memory for reading and writing was 19.5 ns [5].

The performance parameters of NVM was calculated as follows: The leakage power of the NVM cache was assumed to be zero [13], and the performance parameters other than the writing energy, such as reading energy, reading latency, and writing latency, were equal to its corresponding SRAM cache.

To characterize NVM, a writing energy degradation factor p, where p shows how much writing energy is consumed by NVM cache compared to its corresponding SRAM cache, is introduced. For example, it was assumed that a particular SRAM has a writing energy of 1.0 pJ and the NVM cache, which has the same cache parameter as the SRAM cache and a writing energy degradation factor p of 10. The NVM cache has the writing energy consumption of 1.0pJ$\times$10=10pJ. To count the number of cache hits and misses ($N_{I,H,IL1}$, $N_{I,M,IL1}$), a very fast cache configuration simulator was simulated [14].

3.4.2 Energy computation of DL1 cache

E_{DL1} is computed using Eq. (3):

$$\begin{aligned} E_{DL1} = {} & E_{R,DL1} \times N_{R,H,DL1} \\ & + E_{W,DL1} \times N_{R,M,DL} \qquad 1 \\ & + E_{W,DL1} \times (N_{W,H,DL1} + N_{W,M,DL1}) \\ & + P_{leak,DL1} \times T_{total} \end{aligned} \tag{3}$$

where $E_{R,DL1}$ is the reading access energy of DL1 cache, $E_{W,DL1}$ is the writing access energy of DL1, $P_{leak,DL1}$ is the leakage power of DL1, $N_{R,H,DL1}$ is the number of cache hits of data reading access on DL1, $N_{R,M,DL1}$ is the number of cache misses of data reading access on DL1 cache, $N_{W,H,DL1}$ is the number of cache hits of data writing access on DL1, $N_{W,M,DL1}$ is the number of cache misses of data writing access on the DL1 cache, and T_{total} is the total cache access time.

Figs. 3(b) and (c) show the cache energy consumed by the data memory accesses and the required dynamic energy at gray boxes.

In the case of data memory access, a processor core accesses the DL1 cache first. Two types of data memory accesses exist: data reading access and data writing access. Fig. 3(b) shows the cache energy consumed by data reading accesses. In a data reading access, if there is a data reading cache hit on the DL1 cache, the reading energy of DL1 cache is needed to read the data from DL1 cache shown by the first term in Eq. (3). If the data reading cache miss on DL1 cache occurs, the processor core accesses the UL2 cache and the writing energy of DL1 cache is required for data replacement, as shown by the second term in Eq. (3). If we have a cache miss on the UL2 cache, the processor core accesses the main memory and the writing energy of UL2 cache is needed for data replacement. Fig. 3(c) shows the cache energy consumed by data writing accesses. In a data-writing access, the writing energy of DL1 cache is required regardless of the cache hit or miss shown by the third term in Eq. (3) because it is assumed that every level has a write-through function. The static energy of the DL1 cache is not dependent on the memory accesses but it can be calculated by the product of the DL1 leakage power $P_{leak,DL1}$ and the total cache access time T_{total} shown by the fourth term in Eq. (3).

The performance parameters of DL1 cache can be calculated in a similar way as the IL1 cache.

3.4.3 Energy computation of UL2 cache

E_{UL2} is computed by Eq. (4):

$$\begin{aligned} E_{UL2} = {} & E_{R,UL2} \times N_{I,H,UL2} \\ & + E_{W,UL2} \times N_{I,M,UL2} \\ & + E_{R,UL2} \times N_{R,H,UL2} \\ & + E_{W,UL2} \times N_{R,M,UL2} \\ & + E_{W,UL2} \times (N_{W,H,DL1} + N_{W,M,DL1}) \\ & + P_{leak,UL2} \times T_{total} \end{aligned} \tag{4}$$

where $E_{R,UL2}$ is the reading access energy of the UL2 cache, $E_{W,UL2}$ is the writing access energy of UL2, $P_{leak,UL2}$ is the leakage power of UL2, $N_{I,H,UL2}$ is the number of instruction cache hits on UL2, $N_{I,M,UL2}$ is the number of instruction cache misses on the UL2 cache, $N_{R,H,UL2}$ is the number of cache hits of data reading access on UL2, $N_{R,M,UL2}$ is the number of cache misses of data reading access on the UL2 cache, $N_{W,H,DL1}$ is the number of cache hits of data writing access on DL1, $N_{W,M,DL1}$ is the number of cache misses of data writing access on the DL1 cache, and T_{total} is the total cache access time.

In the case of UL2 cache accesses, a cache miss already occurs on the IL1 or DL1 cache. It has similar behavior to IL1 or DL1 cache other than data writing access. Now there are three types of memory access: instruction access, data reading access and data writing access.

In the case of instruction accesses, a processor core accesses the IL1 cache at first, but there is a cache miss on the IL1 cache and the processor core accesses the UL2 cache. Fig. 3(a) shows the cache energy consumed by instruction memory accesses, and dynamic energy is required at the gray boxes in Fig. 3(a). If there is a cache hit on the UL2 cache, the reading energy of the UL2 cache is needed to read the data from the UL2 cache shown by the first term in Eq. (4). If there is a cache miss on the UL2 cache, the processor core accesses the main memory and writing energy of UL2 cache is required for data replacement shown by the second term in Eq. (4).

In the case of data reading access, a processor core accesses the DL1 cache first, but there is a cache miss on the DL1 cache, and the processor core accesses the UL2 cache. Fig. 3(b) shows the cache energy consumed by data reading access and dynamic energy is needed at the gray boxes in Fig. 3(b). If there is a cache hit on the UL2 cache, the reading energy of the UL2 cache is needed to read data from UL2 cache shown by the third term in Eq. (4). If there is a cache miss on the UL2 cache, the processor core accesses the main memory and the writing energy of UL2 cache is needed for data replacement, as shown by the fourth term in Eq. (4).

In the case of the data writing access, a processor core accesses the DL1 cache first, similar to the data reading access. Fig. 3(c) shows the cache energy consumed by data writing accesses and dynamic energy is required at the gray boxes in Fig. 3(c). In data writing access, writing access to the UL2 cache is needed when there is a data writing miss on the DL1 cache. Because it was assumed that every level has a write-through function, writing access to UL2 cache is needed, even when there is a data writing hit on the DL1 cache shown by the fifth term in Eq. (4). The static energy of the UL2 cache is not dependent on memory access but can be calculated by the product of UL2 leakage power, $P_{leak,UL2}$, and the total cache access time T_{total} shown by the sixth term in Eq. (4).

The performance parameters of the UL2 cache can be calculated in a similar way as the IL1 or DL1 cache.

3.4.4 Access time computation

The total cache access time T_{total} can be calculated by summing up the access times of each cache. Eq. (5) shows the total cache access time T_{total}:

$$T_{total} = T_I + T_{D,R} + T_{D,W} \tag{5}$$

where T_I is the total access time required for instruction memory access, $T_{D,R}$ is the total access time required for data reading memory access, and $T_{D,W}$ is the total access time required for data writing memory access.

In a two-level cache, the upper cache is accessed first. If there is a cache hit in the upper cache, the data is retrieved from there. If there is a cache miss in the upper cache, the lower cache or main memory is accessed.

Eq. (6) shows the total access time T_I required for the instruction memory access (see Fig. 4(a)):

$$\begin{aligned} T_I = {} & T_{R,IL1} \times N_{I,H,IL1} \\ & + T_{R,UL2} \times N_{I,H,UL2} \\ & + T_{R,MM} \times N_{I,M,UL2} \end{aligned} \tag{6}$$

where $T_{R,IL1}$ is the single access time for the IL1 cache, $T_{R,UL2}$ is the single access time for the UL2 cache, and $T_{R,MM}$ is the single access time for the main memory. $T_{R,MM}$ = 19.5 ns was used [5].

Eq. (7) shows the access time $T_{D,R}$ needed for data reading memory access (see Fig. 4(b)):

$$\begin{aligned} T_{D,R} = {} & T_{R,DL1} \times N_{R,H,DL1} \\ & + T_{R,UL2} \times N_{R,H,UL2} \\ & + T_{R,MM} \times N_{R,M,UL2} \end{aligned} \tag{7}$$

where $T_{R,DL1}$ is the single reading access time for the DL1 cache, $T_{R,UL2}$ is the single reading access time for the UL2 cache, and $T_{W,MM}$ is the single reading access time for the main memory. $T_{R,MM}$ = 19.5 ns was also used [5].

Eq. (8) shows the total access time $T_{D,W}$ needed for data

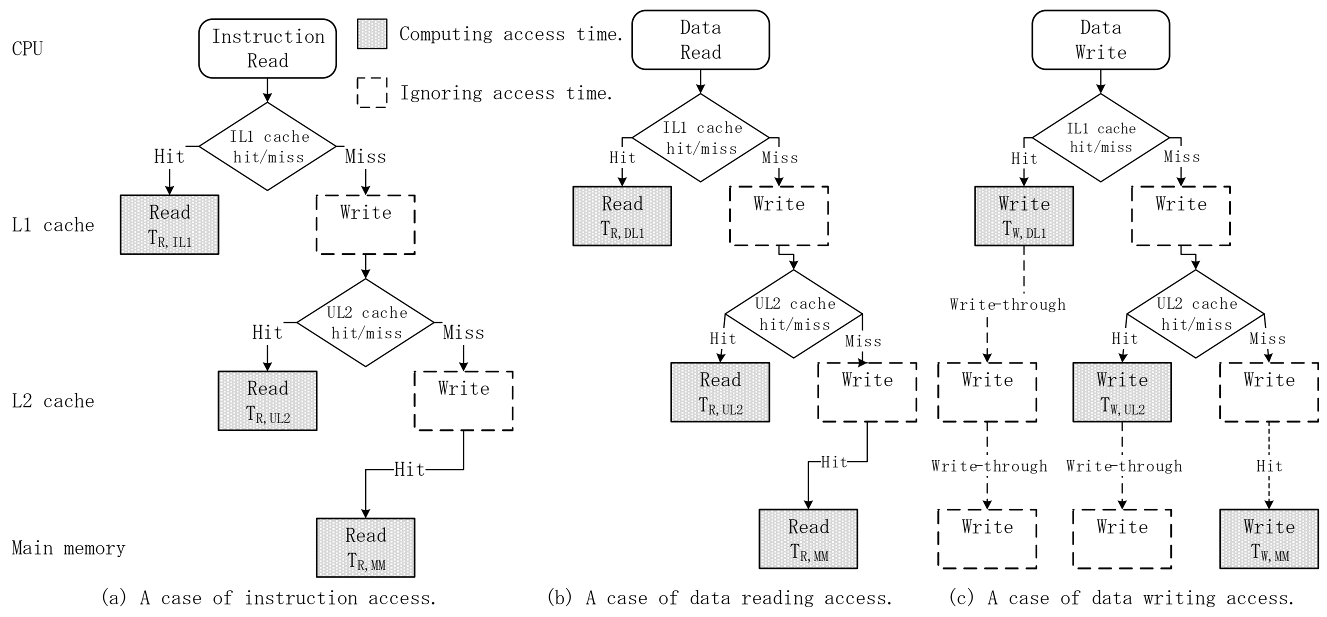


Fig. 4. Access time computation procedure of two-level cache.

writing memory access (see Fig. 4(c)):

$$T_{D,W} = T_{W,DL1} \times N_{W,H,DL1} + T_{W,UL2} \times N_{W,H,UL2} + T_{W,MM} \times N_{W,M,UL2} \tag{8}$$

where $T_{W,DL1}$ is the single writing access time for the DL1 cache, $T_{W,UL2}$ is the single writing access time for the UL2 cache, and $T_{W,MM}$ is the single writing access time for the main memory. $T_{W,MM}$ = 19.5 ns was also used [5].

As shown in Fig. 4, T_{total} shows the total memory access time of running some applications, which must be shorter than the application running time. The static energy of SRAM depends only on its running time not on the cache hits or misses. The larger its running time, the more static energy it consumes. Because NVM cuts off almost all the static energy, NVM will overcome SRAM more if the running time is longer. If T_{total} is used as the lower bound of the application running time and the NVM cache is evaluated, there will be a very fair comparison between SRAM cache and NVM cache.

Note that the time required for write-through was ignored because it was assumed that there is a write buffer for write-through in the access time computation model.

4. Simulations and Evaluations

This section describes two energy evaluations: an energy consumption evaluation for a range of applications, and energy consumption evaluation for when the writing energy of NVM is changed.

4.1 Setup

The energy and access time of our two-level cache was simulated as follows:

(1) Calculate the energy and latency per access for SRAM cache.

(2) Estimate the energy and latency per access for NVM cache.

(3) Obtain the memory access trace data when running an application program.

(4) Count the number of cache hits and misses.

(5) Calculate the total energy of the two-level caches.

Step 1 is to calculate the energy and latency per access for the SRAM cache. The SRAM cache was assumed to have several cache parameters, the energy and latency could then be obtained using CACTI 6.5 [10]. Step 2 is to estimate the energy and latency per access for the NVM cache. The performance parameters of the NVM cache were estimated based on the SRAM cache and the writing energy degradation factor p, as described in Section 3.4.1 (a). Step 3 is to obtain the memory access trace data. Memory access trace data is needed to have cache hits and misses. SimpleScalar/PISA 3.0d [1] was used to obtain memory access trace data for several practical application programs in MediaBench [7]. Step 4 counts the number of cache hits and misses. To calculate the energy and access time of the two-level cache, it is important to count the number of hits and misses for each cache. The fast cache configuration simulator [14] was used to count the cache hits and misses. Step 5 calculates the total energy of the two-level cache. Because sufficient energy, the latency per access of SRAM and NVM (Steps 1 and 2), and the number of cache hits and misses (Step 4) can be obtained, the dynamic energy of the cache can be calculated from the product of energy per access and the number of accesses. The static energy of the cache can be also calculated by the product of the leakage power and the total cache access time. The processes of the calculations above are described in Sections 3.4.1 to 3.4.4. The leakage power can also be determined using CACTI [10].

4.2 Energy Consumption Evaluation for Various Applications

In this experiment, the applications were ADPCM(D), ADPCM(E), EPIC(D), EPIC(E), G721(D), G721(E), JPEG(D), JPEG(E), MPEG2(D) and MPEG2(E) from MediaBench [7]. Based on the Intel ATOM processor [3], seven cache parameters c_1, c_2, c_3, c_4, c_5, c_6, and c_7 were prepared, as shown in Table 1. Cache c_2 has smaller IL1 and DL1 caches than cache c_1, and cache c_3 has larger IL1 and DL1 caches than cache c_1. Cache c_4 has smaller IL1 and DL1 caches and a larger UL2 cache than cache c_1. Cache c_5 has a smaller UL2 cache than cache c_1, and cache c_6 has a larger UL2 cache than cache c_1. Cache c_7 has larger IL1 and DL1 caches, and a smaller UL2 cache than cache c_1.

Using the cache parameter, c_1, as shown in Table 1, and the writing energy degradation factor, p = 10, Fig. 5 compares the energy between the *SRAM-based cache* and the *NVM-based cache (Type I),* and Fig. 6 compares the cache energy between the *SRAM-based cache* and *NVM-based cache (Type II)*. In these figures, the horizontal axes show the name of the application and the vertical axes show the normalized cache energy consumption. As shown in Fig. 5, the static energy of the UL2 cache ranged from 77.3% to 80.8 %, and could be reduced to zero in the *NVM-based cache (Type I)* using NVM instead of SRAM for UL2 cache. In the case of the *NVM-based cache (Type I)*, the dynamic energy of UL2 cache increased approximately 10 times larger than that in the *SRAM-based cache*. In the *NVM-based cache (Type I)*, the energy consumption ranged from 19.1% to 75.4% using NVM for the UL2 cache even when writing energy of NVM was 10 times larger than that of SRAM (i.e. the writing energy degradation factor p = 10). As shown in Fig. 6, static energy and dynamic energy of UL2 cache are the same as the *NVM-based cache (Type I)* described before. The static energy of IL1 cache ranged from 5.2% to 5.4%, and could be reduced to zero in the *NVM-based cache (Type II)* using NVM instead of SRAM for the IL1 and UL2 caches. In the *NVM-based cache (Type II)*, the energy consumption could be reduced from 34.0% to 80.8% as a total using NVM for the IL1 and UL2 caches even when the writing energy of NVM is 10 times larger than that of SRAM (i.e., p = 10).

Using the cache parameter, c_2, as shown in Table 1, and p = 10, Fig. 7 shows the cache energy comparison between the *SRAM-based cache* and *NVM-based cache (Type I)* and Fig. 8 shows the cache energy comparison between the *SRAM-based cache* and *NVM-based cache (Type II)*. As shown in Fig. 7, the static energy of the UL2 cache ranged from 81.5% to 86.1%, and could be reduced to zero in the *NVM-based cache (Type I)* using NVM instead of SRAM for the UL2 cache. In the case of the *NVM-based cache (Type I)*, the dynamic energy of the UL2 cache increased approximately 10 times larger than that in the *SRAM-based cache*. In the *NVM-based cache (Type I)*, the energy consumption could be reduced from 24.2% to 79.5% as a total using NVM for the UL2 cache even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., p = 10). As shown in Fig. 8, the static energy and dynamic energy of the UL2 cache were

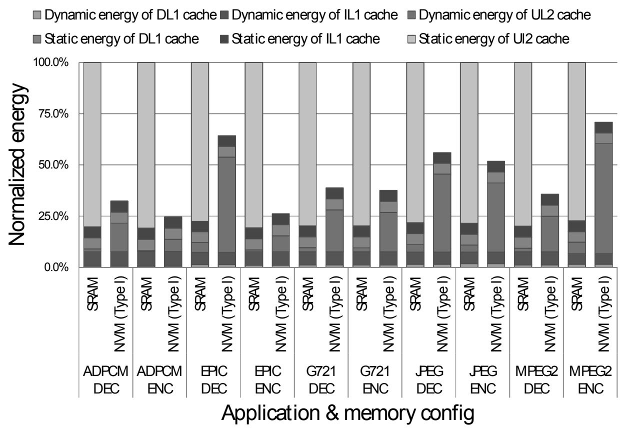


Fig. 5. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type I)* on c_1).

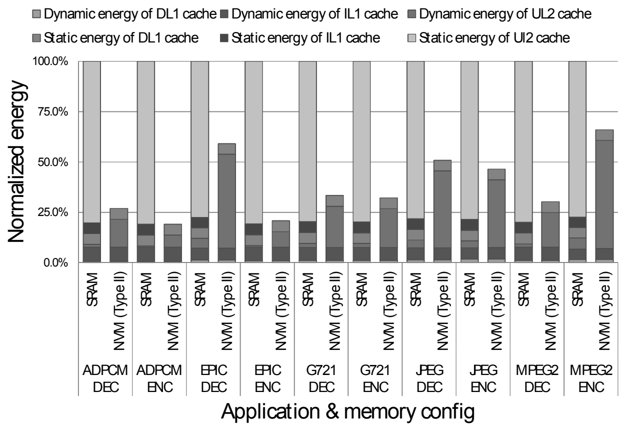


Fig. 6. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type II)* on c_1).

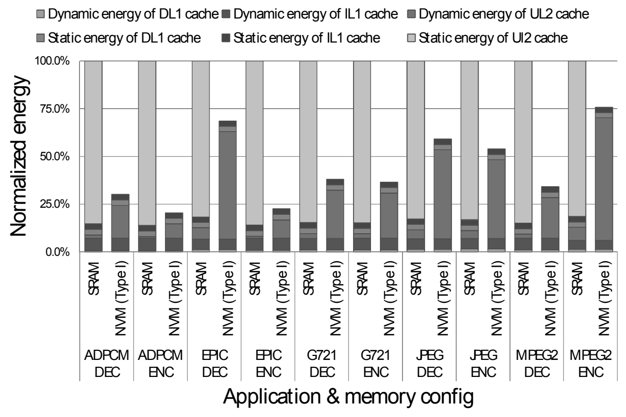


Fig. 7. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type I)* on c_2).

the same as the *NVM-based cache (Type I)* described before. The static energy of the IL1 cache occupies from 2.8% to 2.9%, and can be reduced to zero in the *NVM-based cache (Type II)* using NVM instead of SRAM for the IL1 and UL2 caches. In the *NVM-based cache (Type II)*, the energy consumption could be reduced from 26.4% to 82.4% using NVM for the IL1 and UL2 caches even

Table 1. Experimental Parameters.

Cache parameters	Cache size
c_1 = ((256, 64, 2), (256, 64, 2), (256, 256, 8))	$C(c_1)$ = (32 KB, 32 KB, 512 KB)
c_2 = ((256, 32, 2), (256, 32, 2), (256, 256, 8))	$C(c_2)$ = (16 KB, 16 KB, 512 KB)
c_3 = ((256, 128, 2), (256, 128, 2), (256, 256, 8))	$C(c_3)$ = (64 KB, 64 KB, 512 KB)
c_4 = ((256, 32, 2), (256, 32, 2), (256, 256, 16))	$C(c_4)$ = (16 KB, 16 KB, 1024 KB)
c_5 = ((256, 64, 2), (256, 64, 2), (256, 128, 8))	$C(c_5)$ = (32 KB, 32 KB, 256 KB)
c_6 = ((256, 64, 2), (256, 64, 2), (256, 256, 16))	$C(c_6)$ = (32 KB, 32 KB, 1024 KB)
c_7 = ((256, 128, 2), (256, 128, 2), (256, 128, 8))	$C(c_7)$ = (64 KB, 64 KB, 256 KB)

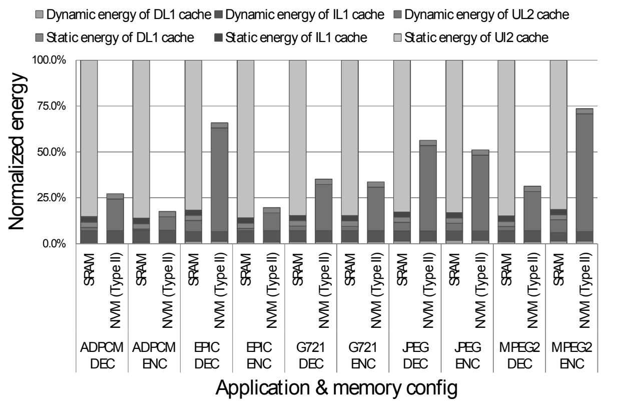


Fig. 8. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type II)* on c_2).

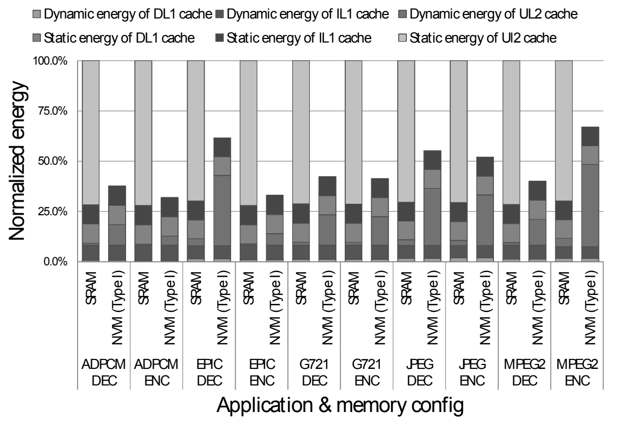


Fig. 9. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type I)* on c_3).

when writing energy of NVM was 10 times larger than that of SRAM (i.e., p = 10).

When the cache parameter c_3, was used, as shown in Table 1 and p = 10, Fig. 9 shows the cache energy comparison between the *SRAM-based cache* and *NVM-based cache (Type I)* and Fig. 10 shows the cache energy comparison between the *SRAM-based cache* and the *NVM-based cache (Type II)*. As shown in Fig. 9, the static energy of the UL2 cache ranged from 69.8% to 72.0%, and could be reduced to zero in the *NVM-based cache (Type I)* using NVM instead of SRAM for UL2 cache. In the case of the *NVM-based cache (Type I)*, the dynamic energy of the UL2 cache increased approximately 10 times larger than that in the *SRAM-based cache*. In the *NVM-based cache (Type I)*, the energy consumption could be reduced from 33.0% to 68.1% using NVM for the UL2 cache, even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., p = 10). As shown in Fig. 10, the static energy and dynamic energy of UL2 cache were the same as the *NVM-based cache (Type I)* described previously. The static energy of IL1 cache ranged from 9.3% to 9.6%, and could be reduced to zero in the *NVM-based cache (Type II)* using NVM instead of SRAM for IL1 and UL2 caches. In the *NVM-based cache (Type II)*, the energy consumption could be reduced from 42.1% to 77.7% as a total using NVM for IL1 and UL2 caches even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., p = 10).

Using the cache parameter c_4, as shown in Table 1, and p = 10, Fig. 11 shows the cache energy comparison between the *SRAM-based cache* and the *NVM-based cache (Type I)*, and Fig. 12 shows the cache energy comparison between the *SRAM-based cache* and the *NVM-based cache (Type II)*. As shown in Fig. 11, the static energy of the UL2 cache ranged from 87.0% to 92.2%, and could be reduced to zero in the *NVM-based cache (Type I)* using NVM instead of SRAM for the UL2 cache. In the case of the *NVM-based cache (Type I)*, the dynamic energy of the UL2 cache increased to approximately 10 times larger than that in the *SRAM-based cache*. In the *NVM-based cache (Type I)*, the energy consumption could be reduced from 30.5% to 85.6% using NVM for the UL2 cache even when the writing energy of NVM was 10 times larger than that of SRAM (i.e. p = 10). As shown in Fig. 12, the static energy and dynamic energy of the UL2 cache were the same as the *NVM-based cache (Type I)* described previously. The static energy of the IL1 cache ranged from 1.5% to 1.6%, and could be reduced to zero in the *NVM-based cache (Type II)* using NVM instead of SRAM for the IL1 and UL2 caches. In the *NVM-based cache (Type II)*, the energy consumption could be reduced from 31.7% to 87.2% as a total using NVM for the IL1 and UL2 caches even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., p = 10).

Using a cache parameter of c_5, as shown in Table 1, and p = 10, Fig. 13 shows the cache energy comparison between the *SRAM-based cache* and *NVM-based cache*

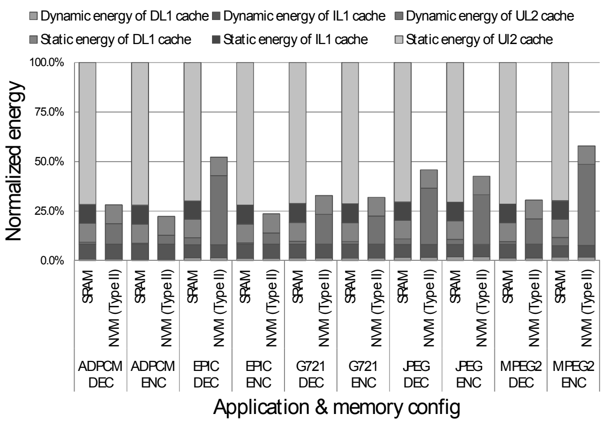


Fig. 10. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type II)* on c_3).

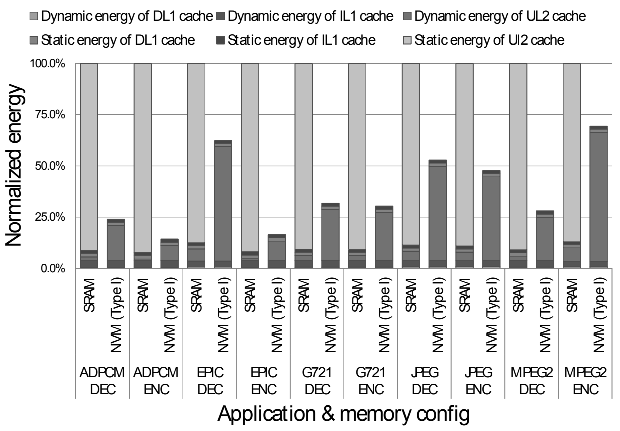


Fig. 11. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type I)* on c_4).

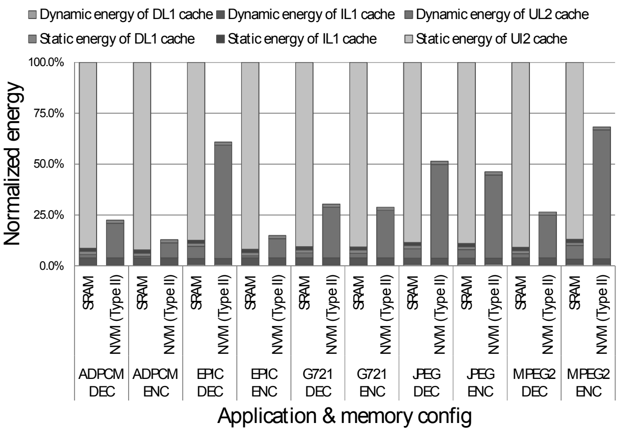


Fig. 12. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type II)* on c_4).

(Type I), and Fig. 14 shows the cache energy comparison between *SRAM-based cache* and the *NVM-based cache (Type II)*. As shown in Fig. 13, the static energy of UL2 cache ranged from 67.1% to 69.1%, and could be reduced to zero in the *NVM-based cache (Type I)* using NVM instead of SRAM for the UL2 cache. In the case of the *NVM-based cache (Type I)*, the dynamic energy of the

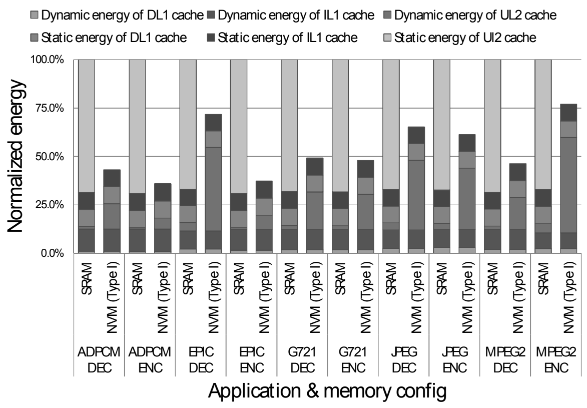


Fig. 13. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type I)* on c_5).

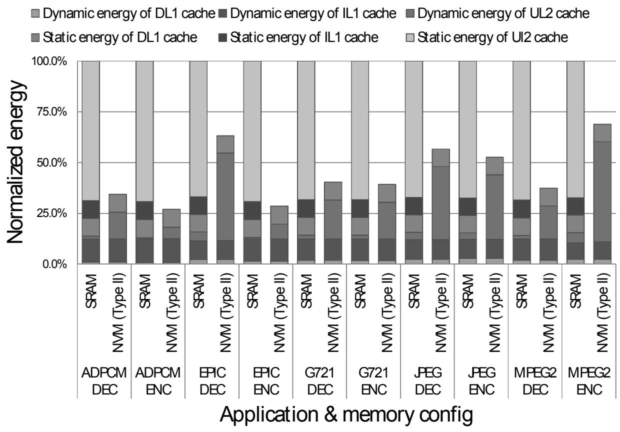


Fig. 14. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type II)* on c_5).

UL2 cache increased approximately 10 times more than that in the *SRAM-based cache*. In the *NVM-based cache (Type I)*, the energy consumption could be reduced from 33.0% to 64.1% using NVM for the UL2 cache even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., p = 10). As shown in Fig. 14, the static energy and the dynamic energy of UL2 cache were the same as the *NVM-based cache (Type I)* described previously. The static energy of the IL1 cache ranged from 8.6% to 8.9%, and could be reduced to zero in the *NVM-based cache (Type II)* using NVM instead of SRAM for the IL1 and UL2 caches. In the *NVM-based cache (Type II)*, the energy consumption could be reduced from 31.1% to 72.9% using NVM for the IL1 and UL2 caches even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., $p = 10$).

Using the cache parameter c_6, as shown in Table 1, and p = 10, Fig. 15 shows the cache energy comparison between the *SRAM-based cache* and the *NVM-based cache (Type I),* and Fig. 16 shows the cache energy comparison between the *SRAM-based cache* and the *NVM-based cache (Type II)*. As shown in Fig. 15, the static energy of UL2 cache ranged from 84.9% to 89.1%, and could be reduced to zero in the *NVM-based cache (Type I)* using NVM instead of SRAM for UL2 cache. In the case of the *NVM-*

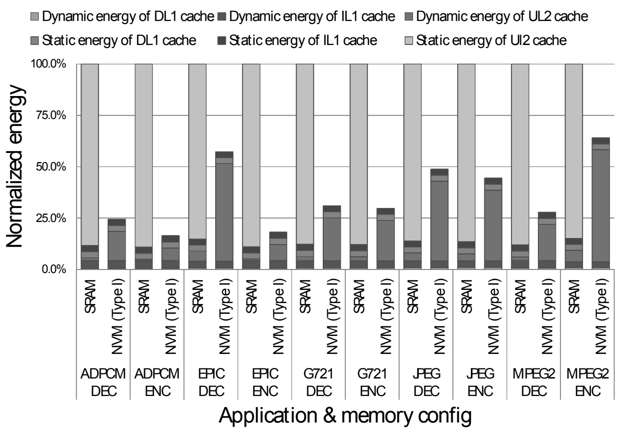


Fig. 15. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type I)* on c_6).

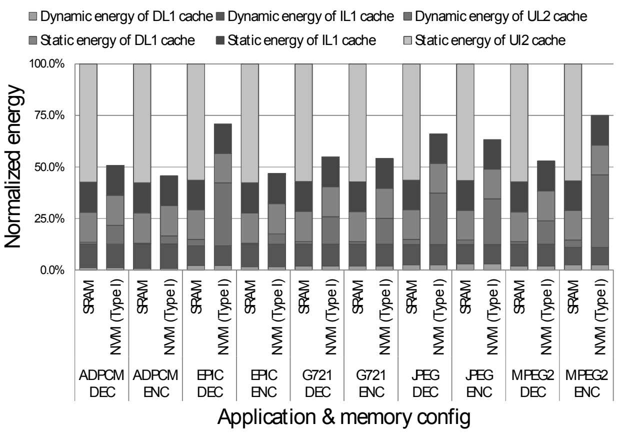


Fig. 17. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type I)* on c_7).

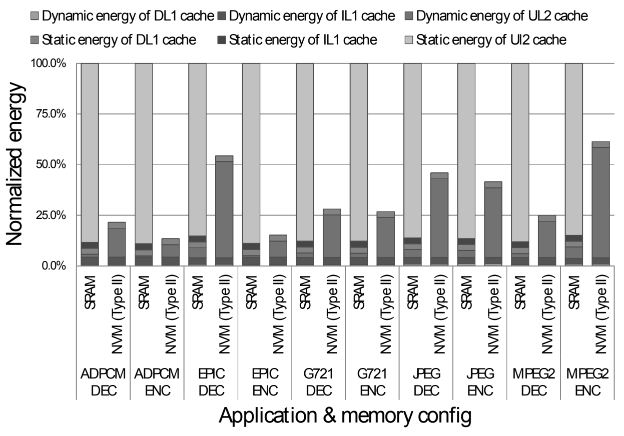


Fig. 16. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type II)* on c_6).

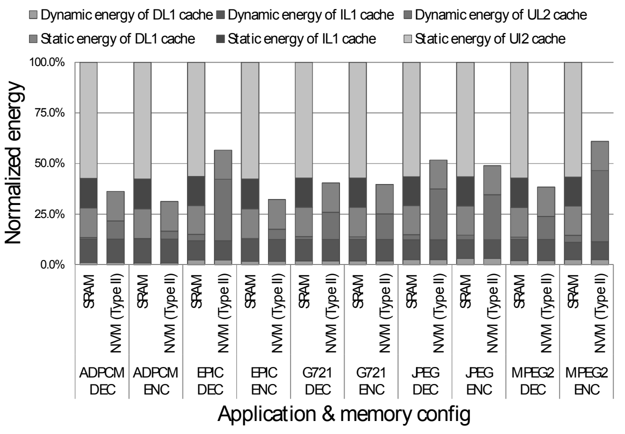


Fig. 18. Energy computation evaluation (*SRAM-based cache* vs. *NVM-based cache (Type II)* on c_7).

based cache (Type I), the dynamic energy of UL2 cache increased approximately 10 times more than that in the *SRAM-based cache*. In the *NVM-based cache (Type I)*, the energy consumption could be reduced from 36.0% to 83.6% using NVM for the UL2 cache even when the writing energy of NVM was 10 times more than that of SRAM (i.e., $p = 10$). As shown in Fig. 16, the static energy and dynamic energy of the UL2 cache were the same as the *NVM-based cache (Type I)* described previously. The static energy of IL1 cache occupies from 2.9% to 3.0%, and it can be reduced to zero in the *NVM-based cache (Type II)* using the NVM instead of SRAM for IL1 and UL2 caches. In the *NVM-based cache (Type II)*, the energy consumption could be reduced from 38.7% to 86.6% as a total using NVM for IL1 and UL2 caches even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., $p = 10$).

Using the cache parameter c_7, as shown in Table 1, and $p = 10$, Fig. 17 compares the cache energy of the *SRAM-based cache* and the *NVM-based cache (Type I)*, and Fig. 18 compares the cache energy of the *SRAM-based cache* and the *NVM-based cache (Type II)*. As shown in Fig. 17, the static energy of the UL2 cache ranged from 56.5% to 57.7%, and could be reduced to zero in the *NVM-based cache (Type I)* using NVM instead of SRAM for UL2 cache. In the case of the *NVM-based cache (Type I)*, the dynamic energy of the UL2 cache increased approximately 10 times more than that in the *SRAM-based cache*. In the *NVM-based cache (Type I)*, the energy consumption could be reduced from 25.0% to 54.2% overall using NVM for the UL2 cache even when the writing energy of NVM was 10 times larger than that of SRAM (i.e. the writing energy degradation factor $p = 10$). As shown in Fig. 18, the static energy and dynamic energy of the UL2 cache were the same as the *NVM-based cache (Type I)* described previously. The static energy of the IL1 cache ranged from 14.3% to 14.6%, and could be reduced to zero in the *NVM-based cache (Type II)* using NVM instead of SRAM for the IL1 and UL2 caches. In the *NVM-based cache (Type II)*, the energy consumption could be reduced from 29.1% to 68.8% using NVM for the IL1 and UL2 caches even when the writing energy of NVM was 10 times larger than that of SRAM (i.e., $p = 10$).

The experimental results suggest that, even if the writing energy of NVM is 10 times larger than SRAM, the NVM-based caches can reduce their energy consumption over the SRAM-based caches.

4.3 Energy Consumption Evaluation Changing NVM Writing Energy

This section shows the energy consumption evaluation when the writing energy degradation factor p which is used to estimate NVM writing energy, is changed. In this experiment, the applications used were ADPCM(E) and MPEG2(E). Based on the Intel ATOM processor [3], the cache parameters c_1 was used, as shown in Table 1.

Figs. 19 and 20 show the cache energy consumption of ADPCM(E) when varying the writing energy degradation factor p. In these figures, the horizontal axes show the writing energy degradation factor p and the vertical axes show the ratio of energy consumption when it was assumed that the energy consumption of both *NVM-based caches (Type I and Type II)* corresponding to their *SRAM-based cache* to be 100%. The static energy of the UL2 cache in the *SRAM-based cache* was 80.8% and could be reduced to zero in both *NVM-based caches (Type I and Type II)*. The static energy of the IL1 cache in the *SRAM-based cache* architecture occupied 5.4% and could be reduced to zero in the *NVM-based cache (Type II)*. On the other hand, as the writing energy degradation factor p increased, the dynamic energy of UL2 cache increased from 0.6% to 6.0% in both *NVM-based caches (Type I and Type II)*. The dynamic energy of the IL1 cache also increased slightly, by 0.6%, but the ratio did not change.

Figs. 20 and 21 show the cache energy consumption of MPEG(E) when varying the writing energy degradation factor p. The static energy of the UL2 cache in the *SRAM-based cache* was 77.3% and could be reduced to zero in both *NVM-based caches (Type I and Type II)*. The static energy of the IL1 cache in the *SRAM-based cache* was 5.4% and could be reduced to zero in the *NVM-based cache (Type II)*. On the other hand, as the writing energy degradation factor p increases, the dynamic energy of UL2 cache increased from 5.6% to 53.7% in both *NVM-based caches (Type I and Type II)*. The dynamic energy of IL1 cache also increased from 5.18% to 5.50%.

Overall, these results revealed the effectiveness of using NVM caches for a two-level cache architecture, even when the writing energy degradation factor p = 10.

5. Conclusion

NVM has some advantages over SRAM, such as a low leakage power or non-volatility, but it incurs high writing

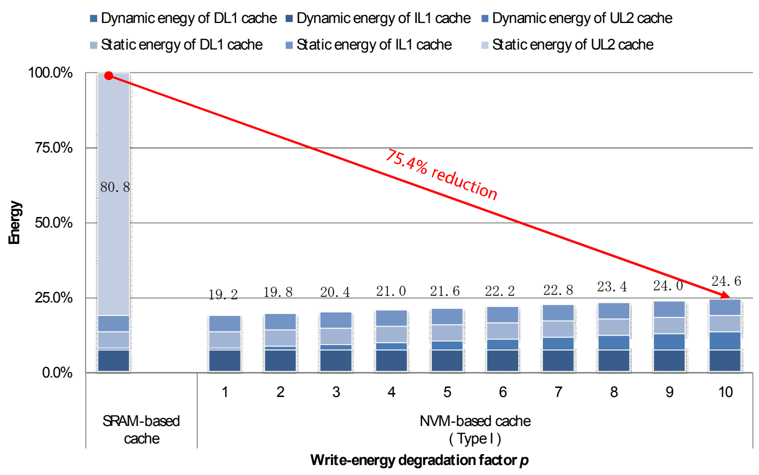


Fig. 19. Energy computation evaluation (ADPCM(E) and *NVM based cache (Type I)* on c_1).

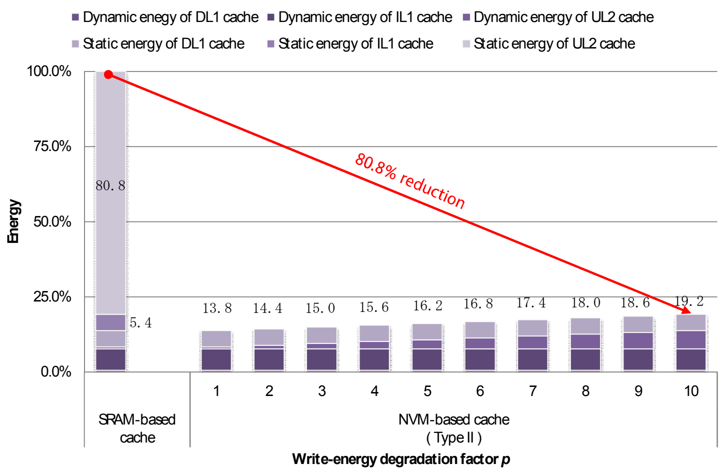


Fig. 20. Energy computation evaluation (ADPCM(E) and *NVM based cache (Type II)* on c_1).

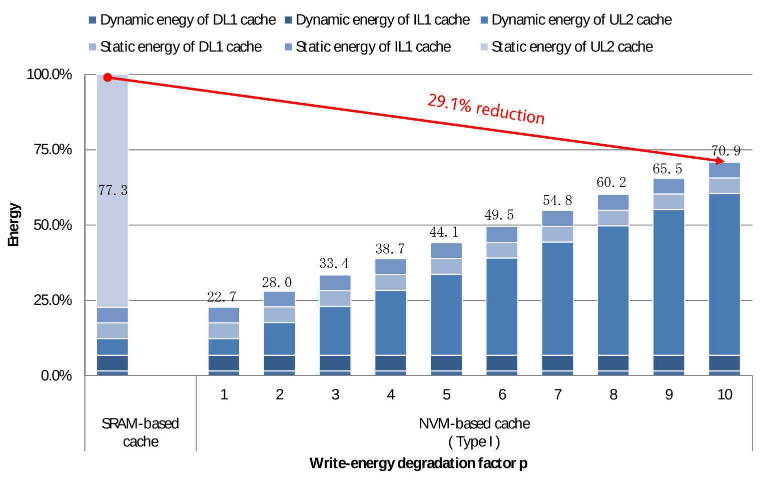


Fig. 21. Energy computation evaluation (MPEG2(E) and *NVM based cache (Type I)* on c_1).

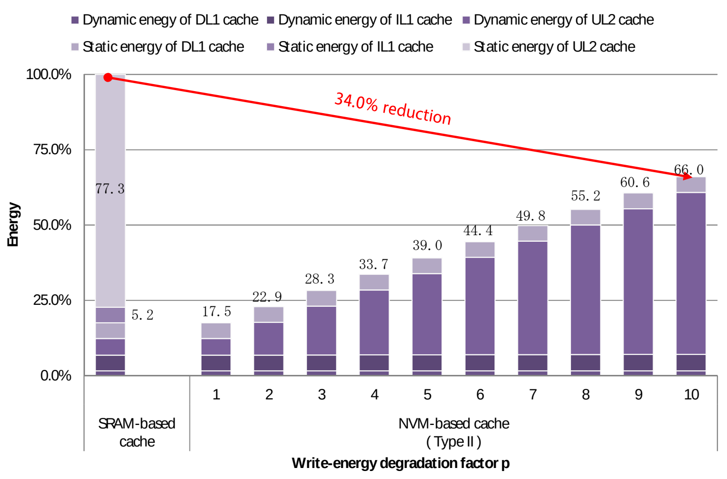


Fig. 22. Energy computation evaluation (MPEG2(E) and *NVM based cache (Type II)* on c_1).

energy. This study assumed a two-level cache using NVM partially and evaluated its energy consumption.

Even if the NVM writing energy is 10 times larger than that of SRAM, the NVM-based caches can reduce their energy consumption over SRAM-based caches. Because embedded systems or mobile systems normally have a large standby time, they can be ignored. NVM is expected to be one of the most significant options for future cache architectures.

NVM-based caches can reduce the high leakage power, but they can increase the dynamic energy of the writing operation. Overall, they are effective in energy saving to use NVM partially in caches.

Acknowledgement

This study is supported by NEC Corporation, Waseda University Grant for Special Research Project (No. 2012A-603), and Grand-in-Aid for Scientific Research (No. 25280017).

References

[1] T. Austin, E. Larson, and D. Ernst, “SimpleScalar: an infrastructure for computer system modeling,” *IEEE Trans. on Computers*, vol. 35, pp. 59-67, Feb. 2002. Article (CrossRef Link)

[2] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, “Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement,” in *Proc. of DAC 2008*, pp. 554-559, June 2008. Article (CrossRef Link)

[3] G. Gerosa, S. Curtis, M. D'Addeo, B. Jiang, B. Kuttanna, F. Merchant, B. Patel, M. Taufique, and H. Samarchi, “A sub-1W to 2W low-power IA processor for mobile internet devices and ultra-mobile PCs in 45nm Hi-k metal gate CMOS,” in *Proc. of ISSCC*, pp. 256-611, Feb. 2008. Article (CrossRef Link)

[4] J. Hu, C. J. Xue, Q. Zhuge, W. Tseng, and E. H.-M. Sha, “Towards energy efficient hybrid on-chip Scratch Pad Memory with non-volatile memory,” in *Proc. of DATE 2011*, pp. 1-6, Mar. 2011. Article (CrossRef Link)

[5] A. Janapsatya, A. lgnjatovic, and S. Parameswaran, "Finding optimal L1 cache configuration for embedded systems," in *Proc. of ASP-DAC*, pp. 796-801, Jan. 2006. Article (CrossRef Link)
[6] Y. Joo and S. Park, "A Hybrid PRAM and STT-RAM Cache Architecture for Extending the Lifetime of PRAM Caches," *IEEE Computer Architecture Letters*, Issue. 99, Aug. 2012. Article (CrossRef Link)
[7] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems," in *Proc. of MICRO-30*, pp. 330-335, Dec. 1997. Article (CrossRef Link)
[8] S. Matsuno, M. Tawada, M. Yanagisawa, S. Kimura, N. Togawa, and T. Sugibayashi, "Evaluation of two-level cache with Non-Volatile Memory for embedded processors," in *Proc. of ITC-CSCC 2013*, pp. 250-253, July 2013.
[9] S. Matsuno, M. Tawada, M. Yanagisawa, S. Kimura, N. Togawa, and T. Sugibayashi, "Energy evaluation for two-level on-chip cache with Non-Volatile Memory on mobile processors," in *Proc. of ASICON 2013*, pp. 31-34, Oct. 2013.
[10] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: a tool to model large caches," http://www.cs.utah.edu/~naveen/cacti_report.pdf, 2009. Article (CrossRef Link)
[11] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *Proc. of HPCA*, pp. 50-61, Feb. 2011. Article (CrossRef Link)
[12] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proc of HPCA*, pp. 239-249, Feb. 2009. Article (CrossRef Link)
[13] H. Sun, C. Liu, W. Xu, J. Zhao, N. Zheng, and T. Zhang, "Using magnetic RAM to build low-power and soft error-resilient L1 cache," *IEEE Trans. VLSI Systems*, Vol. 20, No. 1, pp. 19-28, Dec. 2012. Article (CrossRef Link)
[14] M. Tawada, M. Yanagisawa, T. Ohtsuki, and N. Togawa, "Exact and fast L1 cache configuration simulation for embedded systems with FIFO/PLRU cache replacement policies," in *Proc. of VLSI-DAT*, pp. 1-4, Apr. 2011. Article (CrossRef Link)
[15] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "Energy reduction for STT-RAM using Early Write Termination," in *Proc. of ICCAD*, pp. 264-268, Nov. 2009. Article (CrossRef Link)

Shota Matsuno received his B. Eng. and M. Eng. degree from Waseda University in 2011 and 2013. He is currently working toward D. Eng. degree. His research interests include memory design, non-volatile memory and embedded system. He is a student member of the IEICE of Japan.

Masashi Tawada received his B. Eng., M. Eng. degrees from Waseda University in 2010, and 2012, respectively, all in computer science. He is currently working toward D. Eng. degree there. His research interests are cache design and embedded architecture.

Masao Yanagisawa received his B. Eng., M. Eng. and Dr. Eng. degrees from Waseda University in 1981, 1983 and 1986, respectively, all in electrical engineering. He was with University of California, Berkeley from 1986 through 1987. In 1987, he joined Takushoku University. In 1991, he left Takushoku University and joined Waseda University, where he is currently a Professor in the Department of Computer Science and Engineering. His research interests are combinatorics and graph theory, computational geometry, VLSI design and verification, and network analysis and design. He is a member of IEEE, ACM, IPSJ and IEICE.

Shinji Kimura received his B.E., M.E. and Dr. of engineering degrees in information science from Kyoto University, Kyoto, Japan in 1982, 1984 and 1989, respectively. He was an Assistant Professor at Kobe University from 1985, and an Associate Professor of Nara Institute of Science and Technology from 1983. He has been a Professor of Waseda University since 2002. He was a visiting scientist at Carnegie Mellon University from 1989 to 1990, and was a visiting scholar of Stanford University from 2000 to 2001. He is interested in the formal and timing verification of logic circuits, hardware/software codesign methodologies, reconfigurable hardware, and the low-power design. He is a member of ACM, IEEE, IEICE and IPSJ.

Tadahiko Sugibayashi received his B.S. and M.S. degrees in material science from Osaka University, Osaka, Japan, in 1984 and 1986, respectively. He joined NEC Corporation in 1986, where he worked on memory LSI design. He is now engaged in the development of ultra-low power circuit and systems. He is a senior manager of Green Platform Research Laboratories, NEC Corporation, Ibaraki, Japan.

Nozomu Togawa received his B. Eng., M. Eng. and Dr. Eng. degrees from Waseda University in 1992, 1994 and 1997, respectively, all in electrical engineering. He is currently a Processor in the Department of Computer Science and Engineering, Waseda University. His research interests are VLSI design, graph theory, and computational geometry. He is a member of IEEE, IPSJ and IEICE.