Leveraging Data Lifetime for Energy-Aware Last Level Non-Volatile SRAM Caches using Redundant Store Elimination

Hsiang-Jen Tsai[1], Chien-Chih Chen[1], Keng-Hao Yang[1], Ting-Chin Yang[2], Li-Yue Huang[2], Ching-Hao Chung[2], Meng-Fan Chang[2] and Tien-Fu Chen[1]

[1]Department of CS, National Chiao Tung University, Hsinchu, Taiwan, ROC
{hjtsai, ccchen99, khyang, tfchen}@cs.nctu.edu.tw
[2]Department of EE, National Tsing Hua University, Hsinchu, Taiwan, ROC
{s101061587, s101061575}@m101.nthu.edu.tw, u9661204@oz.nthu.edu.tw, mfchang@ee.nthu.edu.tw

ABSTRACT

NVM has commonly been used to address increasingly large last-level caches (LLCs) requirements by reducing leakage. However, frequent data-writing operations result in increased energy consumption. In this context, a promising memory technology, Non-volatile SRAM (nvSRAM), enables normal and standby operation modes which can be used to store various types of data. However, nvSRAM suffers from high dynamic energy usage due to frequent switching between operation modes. In this paper, we propose a redundant store elimination (RSE) scheme which, on average, discards 94% of needless bit-write operations. Moreover, we present a retention-aware cache management policy to reduce data updates of cache blocks, based on the correlation between data lifetime and cache types. Experimental results demonstrate that our proposal can improve energy consumption of SRAM-based and RRAM-based LLCs by 57% and 31%, respectively.

Categories and Subject Descriptors

B.3.2 [**Hardware**]: Memory Structures—*Cache memories;* B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Advanced technologies, Memory technologies*

General Terms

Design, Management, Experimentation

Keywords

Non-volatile SRAM, Redundant store elimination, Energy reduction, On-chip cache architecture, Memory structure, Non-volatile memory, Multi-core

1. INTRODUCTION

With significant advancements in technology scaling, the performance gap between processor and main memory has driven the demand for on-chip cache memory and it is expected this need will continue to grow significantly. Last-level caches (LLCs) are thus the best choice to bridge the performance and power gap between processor and main memory. Traditionally, SRAM has been used to implement on-chip caches in high-performance processors due to its low operational power requirements and fast read/write latency. However, the disadvantage of SRAM is that it consists of a low density memory cell that has high leakage power.

There exist several potential candidates to replace SRAM with competitive read time, low leakage power consumption, immunity to radiation-induced soft errors and high density features; examples of these include STT-RAM (spin-transfer torque magnetic random access memory) [1] and RRAM (resistive random-access memory) [2]. Their high-density property can provide 4x denser capacity than SRAM in the same surface area with near-zero leakage energy. More importantly, due to their compatibility with CMOS processes they are an attractive option to implement low-power high-density LLCs [3]. Combining SRAM with NVM in LLCs helps reduce energy consumption, thereby improving on the performance documented in prior studies [4][5]. The main focus of prior research in hybrid caching is on how to leverage the advantages of read/write ratio and power consumption, and to try to mitigate the well-known issue of asymmetric access. SRAM-based hybrid cache is in fact normally used to implement LLCs due to its large capacity and low leakage power. Nevertheless, hybrid cache design becomes further complicated when considering the character of asymmetric read/write access with process variations [6][7].

The impact of data lifetime on cache blocks in on-chip caches and the corresponding optimization methods are not well explored at the architectural level. Figure 1 illustrates the definition of data lifetime, i.e. the time from when a block is placed into the LLCs until it is written into the block or leaves the cache. In our observations, we find that the data lifetime of cache blocks, as depicted in Figure 2, is distributed into different data lifetime bins for several workloads. If the data lifetime of cache blocks is distributed randomly, using only one cache type such as SRAM- or NVM-based options, it is not suitable for variable data streams. Consequently, we can distinguish the data lifetime of cache blocks to improve performance and energy.

Furthermore, the non-volatile associative memory cell and architecture was presented to achieve symmetric read/write access and have the non-volatility [8]. Non-volatile SRAM (nvSRAM) integrates SRAM cells and NVM devices, forming a direct bit-to-bit connection in a vertical arrangement within a single cell [9]. This setup enables symmetric read/write access and fast parallel data transfer in systems. More importantly, it is able to operate in normal mode (using SRAM) or standby (using RRAM) to achieve better performance and reduce energy consumption with application behavior.

DAC '14, June 01 - 05 2014, San Francisco, CA, USA

http://dx.doi.org/10.1145/2593069.2593153

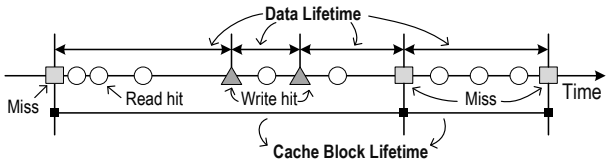


Figure 1: Data lifetime of cache block in a reference stream

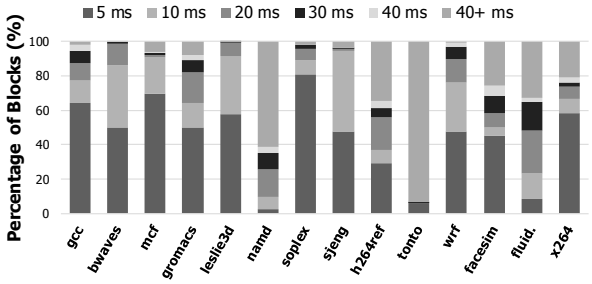


Figure 2: Distribution of blocks in different data lifetime

Based on the above observations, we propose an nvSRAM-based cache architecture employing a sophisticated management policy for cache hierarchy. A key challenge of this design lies in determining a suitable operational mode for the nvSRAM. There is a need to balance the reduced leakage of SRAM cells with cache blocks bearing short data lifetime against the overhead for storing cache blocks with long data lifetime to RRAM. In this work, we present circuit/architecture co-design techniques that exploit the features of nvSRAM to reduce energy consumption, not only in standby mode, but also during program execution. We have addressed several design issues, including what is the relationship between data lifetime and cache types, how to decide an appropriate operational mode for the LLCs, and how we design the cache hierarchy using nvSRAM. In summary, the primary contributions of this paper are as follows:

- We formulate the relationship between data lifetime of cache block and cache types based on SRAM and NVM. We also perform a detailed comparison of caches using different genres of memory technology and we differentiate on the basis of performance, energy consumption and density. This additional contribution is based on our prior research on SRAM, RRAM and nvSRAM bit cells [2][9].
- We propose an energy-efficient cache architecture utilizing nvSRAM features that reduce unnecessary data write operations on bit cells, thereby enabling dynamic energy reduction. This technique does not alter data flow during program execution and hence does not cause any extra cache misses.
- We propose a retention-aware cache management to significantly reduce data updates of cache blocks and utilize a sequential tag-data access scheme to mitigate cache block access conflicts for different applications. These techniques do not require significant architectural modification.
- We employ a device/circuit/architecture co-design to demonstrate that our proposed method is an attractive option for the LLCs as it enables symmetric access, low dynamic energy and low standby power.

2. NON-VOLATILE CACHE DESIGN

In this section, we describe first the basic functions of nvSRAM including read, write, store and restore operations. We next make a case for the use of nvSRAM in LLCs architectures to analyze the differences in SRAM, RRAM and nvSRAM in terms of energy, performance and density. Finally, we explore the relationship between data lifetime and cache types and identify appropriate cache types for different data lifetime.

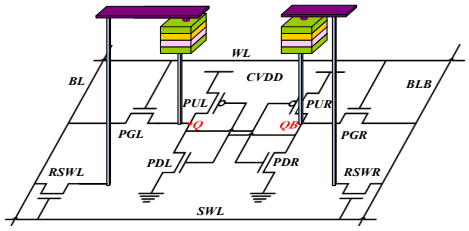


Figure 3: Schematic of non-volatile SRAM cell [9]

2.1 Non-Volatile SRAM Preliminaries

Figure 3 depicts the circuit scheme for a resistive memory based nvSRAM comprising a 6T SRAM cell, 2T RRAM-switch (RSWL and RWSR), and two resistive devices in one cell. The nvSRAM cell inherits all of the advantages of 6T SRAM including low operating energy, symmetric and fast read/write. In the cell structure two resistive devices are vertical-stacked above the 8T SRAM cell and connected directly to SRAM storage nodes (Q and QB) to enable storage of complementary backup data due to non-volatile characteristics. In order to reduce area overhead, the nvSRAM cell does not require an additional control-line (CL) to perform store operations like other nvSRAM cells that share bit lines (BL) with the NVM control-line.

The nvSRAM is able to operate in two modes, normal and standby. In normal mode (or SRAM mode), the features of the nvSRAM cell are identical with SRAM in that it allows fast read/write operations and low dynamic energy consumption. When the cell needs to switch to standby mode, it proceeds to run the store operation (SET and RESET) to flush the SRAM data into the two memristor devices. After the data is successfully backed up, the cell can be completely shut down to eliminate standby leakage. When the cell needs to switch to normal mode, the data of the two memristor devices (RL and RR) are stored to the SRAM storage nodes using a restore operation.

2.2 nvSRAM vs. SRAM and RRAM

In this section, we analyze the reliability, read/write latency, dynamic/static energy consumption and density of the LLCs by comparing different cache types. Traditionally, tags and data arrays in on-chip caches are implemented using SRAM. However, frequent use and fast update are key points in tag arrays for which RRAM or nvSRAM implementations may not be suitable [1][10]. Table 1 lists important quantitative features of three memory technologies: SRAM [9], RRAM [2] and nvSRAM [9]. Several observations may be made from Table 1:

- The SRAM cache and the SRAM component of the nvSRAM cache have symmetric read/write access properties in terms of latency and energy consumption.
- The RRAM cache and the RRAM component of the nvSRAM cache have very different read and write/store properties in terms of latency and energy consumption, with particularly high write/store latency and energy.
- The nvSRAM cache in standby mode is identical to the RRAM cache, both having low leakage power due to their non-volatile property. On the other hand, nvSRAM in normal mode also has high leakage power like SRAM due to the fact that the SRAM component of nvSRAM is active.
- The restore operation overhead in the nvSRAM cache is insignificant due to the features of the cell structure.

Table 1: Comparison of SRAM, RRAM and nvSRAM

		SRAM [9]	RRAM [2]	nvSRAM [9]
Non-volatility			√	√
Symmetric R/W		√		√
Latency	Read	1x	1.51x	1x
	Write	1x	1.94x	1x
	Store	N/A		3.88x
	Restore			0.06x
Energy	Read	1x	1.54x	1.03x
	Write	1x	5.18x	1.03x
	Store	N/A		9.76x
	Restore			0.041x
Leakage	Normal	5.71x	1x	6.11x
	Standby			1x
Density		1.3x	4x	1x

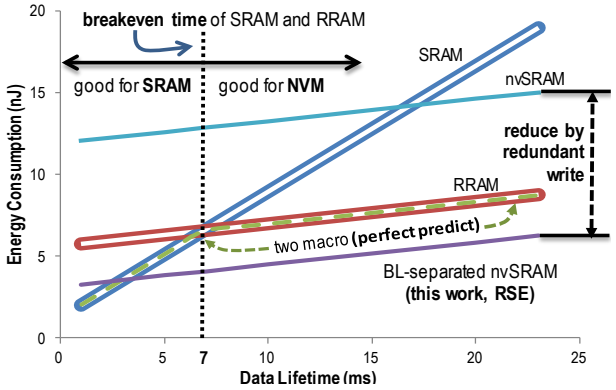


Figure 4: Energy consumption of cache block update as a function of different data lifetime

- The high-density property of RRAM cache can provide 4 times more capacity than SRAM and nvSRAM cache with the same chip area, but it also has the slowest speed.

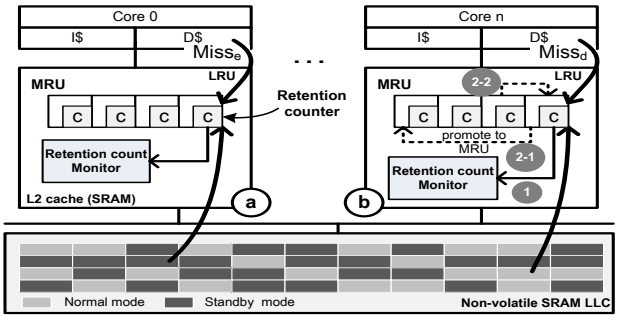
Considering these observations, we note that nvSRAM inherits all of the advantages of SRAM and RRAM, which include fast and symmetric read/write latency, low read/write energy and low leakage in standby mode. On the other hand, it also inherits all of the disadvantages of SRAM and RRAM, such as slow store latency, high store energy, high leakage in normal mode, and especially the low density of SRAM.

2.3 Why nvSRAM?

The high leakage of SRAM increases energy consumption significantly during program execution. On the other hand, the bulk of energy consumption in RRAM is determined by the number of cache access operations and this is due to the high costs of the write operation. The energy consumption for n read operations during the data lifetime period is estimated in Equation 1, where E is the energy consumption for the data lifetime, and L is leakage. E_R, E_W, E_S, E_{RS} are the energy consumption for LLCs read, write, store and restore operations, respectively. Only E_S and E_{RS} occur during standby mode in nvSRAM.

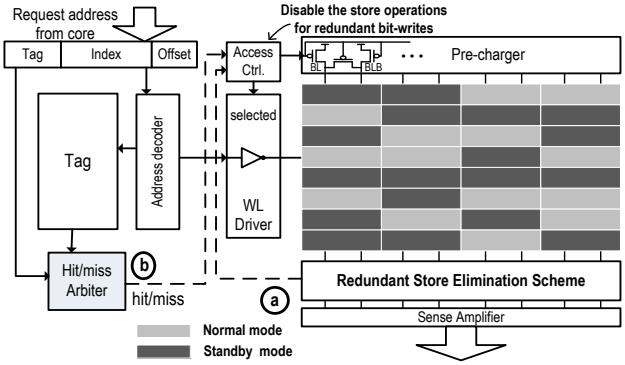
$$E = L + E_W * 1 + E_R * n + E_{RS} * n + E_S * 1 \quad (1)$$

The energy consumption of cache block updates is plotted in Figure 4 as a function of data lifetime for various memory technologies. We observe that 7ms is the breakeven time for SRAM and RRAM. When the data lifetime of the cache block is less than 7ms, the data block should be placed in SRAM-based cache to reduce dynamic energy of the RRAM-based cache. In contrast, when the data lifetime of the cache block exceeds 7ms, the data block should be placed in the RRAM-based cache to reduce



(a) The retention counter of victim block equals zero,
(b) The retention counter of victim block does not equal zero

Figure 5: System architecture with retention-aware policy



(a) Redundant bit-writes discarding by RSE scheme, (b) Avoid access conflicts based on Sequential tag-data access

Figure 6: A non-volatile SRAM last-level cache

leakage power from the SRAM-based cache. Therefore, using only one cache type for different cache blocks is impractical.

The simplest method to deal with a cache block with variable data lifetime is to use two macro design caches (SRAM+RRAM) such that data are written to the appropriate cache type as per their data lifetime using perfect predict. However, the shortcoming of this design is that data lifetime is distributed randomly depending on program behaviors and it is very difficult to precisely predict which cache type the data should be placed in.

As mentioned previously, every cache block has its own data lifetime. Furthermore, every individual cache block have a variable data lifetime depending on the program execution. According to these factors, nvSRAM is an attractive option for LLCs with all of the benefits form SRAM and RRAM. In order to eliminate high leakage power in nvSRAM LLCs, we choose the nvSRAM cell which is always operated in standby mode. Hence, the write operation (data update) in nvSRAM cache consists of two continuous operations i.e. the write operation in SRAM and the store operation in RRAM. However, when nvSRAM is switching to standby mode in each data update, the cell undertakes the store operation in which it flushes SRAM data into two memristor devices which increases dynamic energy significantly. We therefore propose a novel technique, Redundant Store Elimination (RSE), to mitigate the energy overhead of the store operation. Furthermore, we require a new nvSRAM cache management policy to avoid the performance degradation caused by nvSRAM and to reduce the data updates of the LLCs.

3. ENERGY-EFFICIENT NON-VOLATILE SRAM CACHE ARCHITECTURE

Distinct advantages of nvSRAM over SRAM include non-volatility and the non-destructive read ability of the cell. Given nvSRAM, we further propose an energy-efficient cache architec-

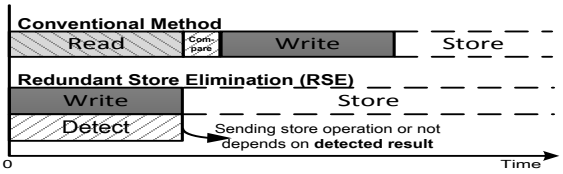
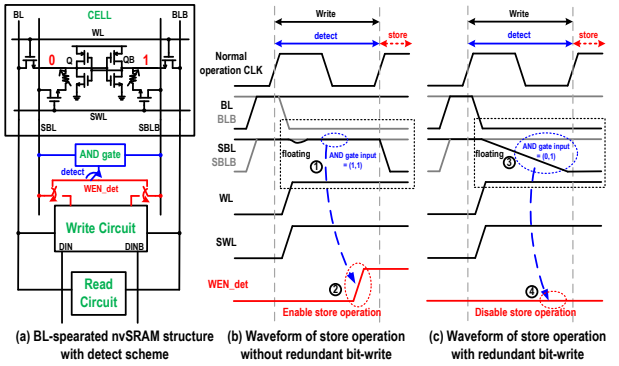


Figure 7: Comparison of the conventional and RSE schemes



(a) BL-spearated nvSRAM structure with detect scheme (b) Waveform of store operation without redundant bit-write (c) Waveform of store operation with redundant bit-write

Figure 8: Redundant store elimination scheme

ture to realize the non-volatile cache hierarchy. In our system architecture, we modify the cache management policy of L2 and cache architecture of the LLCs in the cache hierarchy.

Figure 5(a) illustrates how a retention-aware cache management policy may operate by searching for the victim block with retention counter equals zero in the L2 cache. When the retention counter of victim block does not equal zero ①, the block will be promote to MRU position and decrease the counter until the retention counter of LRU block equals zero ② as shown in Figure 5(b). The more detail replacement algorithm will discuss in Section 3.2. Figure 6(a) presents a schematic of our RSE scheme, implemented in the LLCs to disable the store operation for redundant bit-writes. We utilize sequential tag-data access to reduce cache access conflict in the LLCs as indicated in Figure 6(b). The key feature of this approach is how it leverages the data lifetime of the block in the LLCs, depending on the unpredictable lifetime of the actual data.

3.1 Redundant Store Elimination (RSE)

Prior research shows that we can write data into a cache or memory location and there is a high probability that it does not change the content in the cache [11][12]. We find that on average about 94% of bit-writes are redundant in our experimental results and can be skipped without changing the original stored value. Due to this significant amount of needless bit-writes, there is therefore great potential for energy reduction in the nvSRAM cache. To reduce dynamic energy, we need to ascertain if the new bit data differs from the old.

A conventional method is to first read out the cache content, compare with the new values, and write back the different bits only. However, in this method, every write operation requires first an extra read operation and this increases performance overhead. Alternatively, the "dirty bit" in the higher level cache is used to terminate redundant writes into the lower level cache [1]. This improvement, however, depends on the granularity of block partitions. More importantly, it does not avoid all needless bit-write operations. Since the sub-block in the higher level cache is dirty many bits in it may still be the same.

Based on the above methods, Early Write Termination (EWT) reduces redundant bit-writes effectively at the bit-level [12]. Compared to conventional methods, we are therefore able to do

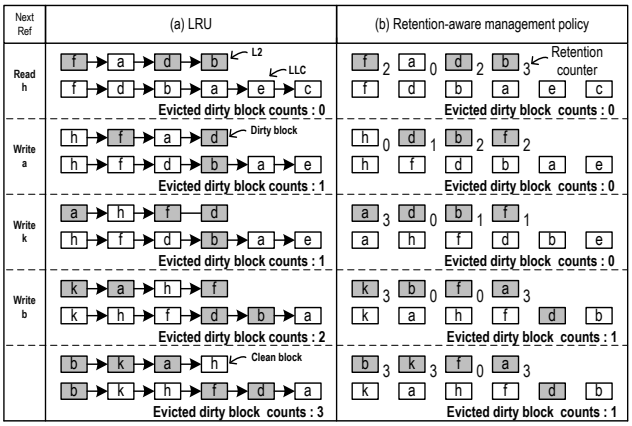


Figure 9: Retention-aware management policy (in order to reduce block update in LLCs, we keep dirty blocks in L2)

better without impacting performance. Our solution is based on the following unique properties of nvSRAM cells:

1. Due to the cell structure of nvSRAM, when updating a cell, we need to first write to SRAM and then store to RRAM. The RRAM cell still maintains its valid previous data in the first stage of a data update operation.
2. The data bit in the SRAM and RRAM components may be different due to the properties of nvSRAM. This means that we can write the new data bit into SRAM and use it to compare with the older data bit from RRAM.
3. In the nvSRAM cell structure, we can read data from two paths (PGL and PGR or RSWL and RSWR). This presents a significant opportunity to minimize latency by writing new data bits to SRAM and comparing it with old data bits at the same time. We can therefore eliminate the extra store operation as soon as redundancy is detected.

Considering these observations, we propose a new detect scheme, Redundant Store Elimination (RSE), for writing a new data bit into SRAM and comparing it with the old data bit without incurring extra performance overhead, as depicted in Figure 7. Figure 8(a) depicts separate bit lines from SRAM and RRAM components in the cell that are used to compare the data bits and generate the control signals. This design makes it possible to write a new data bit to SRAM and compare it with the older data bit in RRAM concurrency. In RSE scheme, we use a AND gate to detect whether the new data bit and old data bit is the same. When a write data is applied to bit lines, WL and SWL enable writing the new data bit into SRAM and reading the old data bit from RRAM. After the old data bit is read from SBL and SBLB, it will input to the AND gate to determine whether to enable or disable the store operation. In Figure 8(c), if the detect result is zero ③, a control signal WEN-det is generated to terminate the store operation on this cell ④. Otherwise, store operation on this cell continues normally ① ② as shown in Figure 8(b).

In summary, RSE detects the older data bit alongside the write operation without an extra read operation in front of the write step. However, the new cell structure of RSE has additional bit lines, necessitating a 10% area overhead compared to the original nvSRAM in layout. The tradeoff is that RSE enables the elimination of all redundant store operations.

3.2 Retention-Aware Management Policy

To reduce the data updates of cache blocks, we propose a new cache management policy to coalesce write-back operations from L2 to LLCs. The key insight here is to prevent the data

Table 2: Simulation platform

CPU model		4-core, 1GHz, out-of-order
Cache simulator		Multi2Sim
SRAM L1 cache		Private, 32KB/32KB, 4 ways, 64B block, write back, 2-cycle R/W
SRAM L2 cache		Private, 256KB, 8 ways, 64B block, write back, 3-cycle R/W
L3 cache	Common	Shared, 8MB, 16 ways, 64 B block, write back
	SRAM	12-cycle R/W
	RRAM	16-cycle read, 20-cycle write
	nvSRAM	12-cycle R/W, 1-cycle restore, 40-cycle store
Main memory		256-entry write buffer, 133-cycle

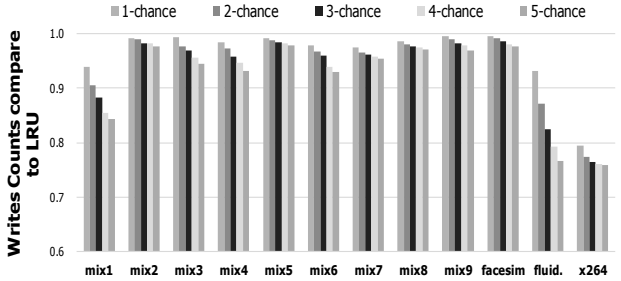


Figure 10: Retention count impact on writes counts

block from premature eviction to the LLCs by accommodating the writes in the dirty block. This is done to increase the residency of the block in L2 and to reduce write-back access in LLCs.

In LRU replacement policy, the LRU block is the candidate for eviction and replacement. In our proposed replacement algorithm, dirty blocks operate differently. We define a retention counter which serves as the retention count of each cache block. The retention counter of the cache block in L2 determines eviction and replacement operations for the block. When the cache has a miss, the LRU block is the first candidate to be checked to see if the block should be evicted from the L2 or not. If the retention counter of the victim block equals zero, the proposed algorithm makes space for the incoming block by evicting this block. However, if the retention counter of the victim block does not equal zero, the proposed algorithm extends the cache block lifetime in the L2 through promoting the block to MRU position and retention counter is decremented. A new victim block is then chosen and this process is repeated until the retention counter for the victim equals zero. In our proposed algorithm, the victim selection process is invisible due to memory latency.

Figure 9(b) describes the behavior of our propose algorithm. When the reference to 'h' misses in L2, the retention counter of the LRU block, 'b', is queried to determine whether the first potential L2 victim can be evicted. When the cache controller responds that the retention counter of the block does not equal to zero, cache block 'b' is promoted to MRU, the retention counter is decremented, and sends a request to probe the next victim candidate 'd'. Similarly, since the retention counter of cache block 'd' does not equal zero, 'd' is promote to MRU and decreases the retention counter. Then, a request to query for the next victim candidate 'a' is dispatched. Since the retention counter of 'a' equals 0, the cache controller allows 'a' to be replaced in the LLCs. Furthermore, when 'b' is re-referenced, it hits in the L2 cache not in the LLCs and the retention count of retention counter in ''b'' is reset to the initial value. The proposed algorithm prevents 'b' form write-back access being in dirty block and reduces write-back access in LLCs effectively as shown in Figure 9(b). Compared to LRU, retention-aware cache manage-

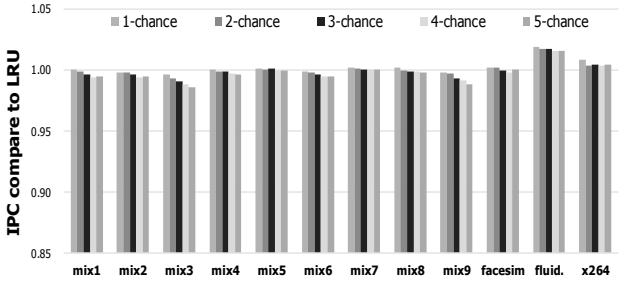


Figure 11: Retention count impact on IPC

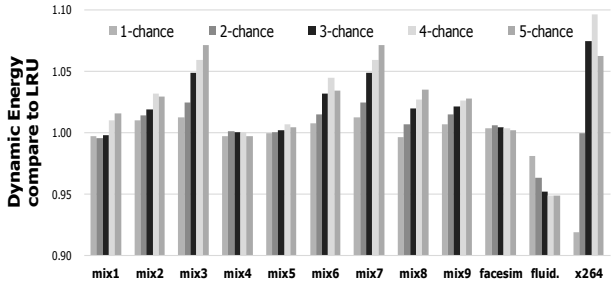


Figure 12: Retention count impact on dynamic energy

ment policy prevents evicting blocks creating write-back access to the LLCs. This is achieved by preventing "dirty block" from becoming victim block in L2.

4. EXPERIMENTAL RESULTS

In this experiment, we implement our design via Multi2Sim simulator [13] and modify the LLCs timing model for asymmetric cache read/write latency operations and a sophisticated management policy. For this experiment, we construct 4 cores with a shared LLCs system as the MOESI directory cache-coherency protocol, and collect the latency/energy results for comparison. Table 2 provides parameters of the simulation platform, where the L1 and L2 caches are private to each core and the shared LLCs is 8MB. All cache module parameters such as read/write latency and energy are derived from our prior research [2][9]. We evaluated the multithreaded benchmark, PARSEC, and multi-program workload mixed with SPEC 2006 in our design. We measure the results by warming up the caches for 400M instructions, and then we report the results for 2B instructions.

4.1 Performance Evaluation

One challenge in implementing nvSRAM cache may be to identify at run-time the latency overhead of store operations that negatively impact system performance. When the latency of the store operation is high it will stall cache access in parallel tag-data access. Sequential tag-data access is a well-known technique employed in LLCs to reduce energy consumption [1]. In sequential tag-data access, a hit or miss of the cache depends on the result of the queries tag array. As a result, significant access conflict may be avoided and energy saving on the data array can be achieved. Overall cache access latency increases when using sequential tag-data access. However, the access latency of the tag array is much smaller than that of the data array due to its smaller size. Therefore, the latency increase in LLCs does not have a significant impact on performance degradation. On average, less than 2% IPC (instructions per cycle) reduction when running the workloads using a 8MB LLCs.

4.2 Retention Count Evaluation

Figure 10, 11 and 12 present the comparison of writes counts, IPC and dynamic energy in LLCs between the retention-aware

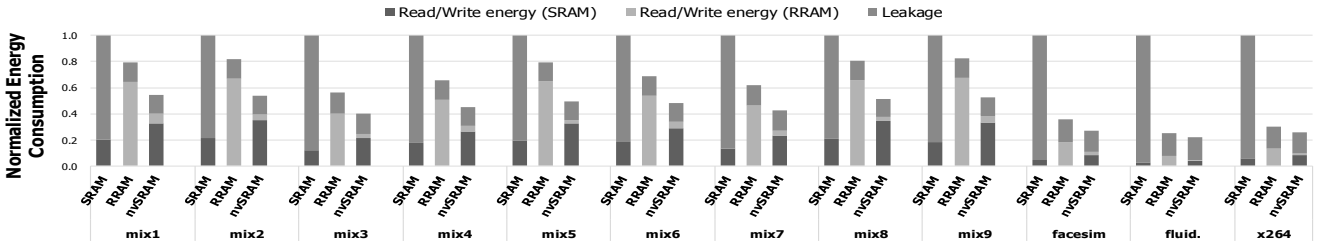


Figure 13: The energy consumption of LLCs on SRAM, RRAM and nvSRAM (normalized to the value of SRAM LLCs)

cache management policy and the LRU policy. Figure 10 shows that when the retention count is increased, the number of writes will decrease, on the contrary, the number of reads will increase. However, the performance and energy gains do not show the same behavior as shown in Figure 11 and 12. As a result, the retention-aware cache management policy with retention count is '2' can get most of the gains in the three metrics.

4.3 Last-Level Cache Energy Evaluation

This work makes nvSRAM a viable alternative for LLCs compared to SRAM and RRAM. Figure 13 shows that the energy consumption is composed of leakage, read/write energy (SRAM) and read/write energy (RRAM) for each cache type. Compared to SRAM and RRAM LLCs with the same capacity, on average, the total energy reduction of a 8MB nvSRAM can save around 57% and 31%, respectively. In the access insensitive workloads, the RRAM and nvSRAM LLCs have good energy consumption because low leakage and small access count in LLCs such as facesim, fluid. and x264. In contrast, the read/write SRAM energy of nvSRAM in access sensitive workloads is increased obviously by compare energy of RSE scheme and read accesses increases due to our proposed management policy that throttle read/write operations. The energy improvement of nvSRAM cache is due to the granularity of data updates is in bit level with the RSE scheme. Therefore, the read/write energy of RRAM is dramatically decreased while this work can significantly reduce store operations of nvSRAM cache.

5. RELATTED WORK

Several studies have explored energy and performance issues for general purpose processors by employing characteristics of NVM caches. Jaleel et al. [14] evaluated performance improvement of Re-reference Interval Prediction (RRIP) by categorizing cache blocks as near re-reference, distant re-reference and long re-reference interval blocks for prevents. RRIP prevents a near re-reference interval cache blocks from premature eviction when choosing an eviction victim. Rasquinha et al. [10] presented STT-RAM cache management policy base on increasing the residency of dirty blocks that prevent the block from being prematurely evicted to higher level caches. However, write-biasing can result in a noticeable performance penalty in L1 due to an increase in the read miss rate in L1. Wang et al. [15] used an obstruction-aware cache management policy to prevent the long write latency of STT-RAM to obstruct the cache port and harm performance of process running in CMPs. Ferreira et al. [16] presented a page replacement policy that reduces PRAM updates in the hybrid DRAM/PRAM main memory by keeping dirty data in DRAM cache and choosing an eviction victim with different priority of clean/dirty pages. Park et al. [17] explored the asymmetric penalty of flash memory read/write operations to choose a clean page as a victim rather than dirty pages in a replacement algorithm. Prior studies of NVM cache have focused on how to reduce energy consumption, and mitigate the impact of asymmetric access. However, our work fundamentally tackles the asymmetric access problem and focuses on how to improve cache management policy for nvSRAM cache.

6. CONCLUSION

This work proposed an energy-efficient nvSRAM cache architecture base on the detail analysis of memory cell features to correlation data lifetime and cache types. We show that nvSRAM cache can have low dynamic energy, symmetric read/write latency and low leakage through switching to the appropriate operation mode. We propose a RSE scheme to discard 94% of needless store operations on average and have a retention-aware cache management policy to reduce write-back access in advance. In our experimental results, nvSRAM can improve more than 57% saving in cache energy compares to SRAM-based cache. We also demonstrate that nvSRAM becomes an attractive option for designing energy-efficient LLCs in CMPs.

7. REFERENCES

[1] S. Park et al., “Future Cache Design using STT MRAMs for Improved Energy Efficiency: Devices, Circuits and Architecture,” in *Proc. DAC*, 2012, pp. 492-497.

[2] S.-S. Sheu et al., “4 Mb Embedded SLC Resistive-RAM Macro with 7.2 ns Read-Write Random-Access Time and 160 ns MLC-Access Capability,” in *Proc. ISSCC*, 2011, pp. 200-202.

[3] K. Lee et al., “Development of Embedded STT-MRAM for Mobile System-On-Chips,” *IEEE Trans. Magn.*, vol. 47, 2011.

[4] X. Wu et al., “Hybrid Cache Architecture with Disparate Memory Technologies," in *Proc. ISCA*, 2009, pp. 34-45.

[5] G. Sun et al., “A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs,” in *Proc. HPCA*, 2009, pp. 239-249.

[6] J. Wang et al., “Point and Discard: A Hard-Error-Tolerant Architecture for Non-Volatile Last Level Caches,” in *Proc. DAC*, 2012.

[7] Y. Zhoy et al., “Asymmetric-access aware Optimization for STT-RAM Caches with Process Variations,” in *Proc. GLSVLSI*, 2013.

[8] Y. Ma et al., “An MTJ-Based Nonvolatile Associative Memory Architecture With Intelligent Power-Saving Scheme for High-Speed Low-Power Recognition Applications,” in *Proc. ISCAS*, 2013, pp. 1248-1251.

[9] P.-F. Chiu et al., “Low Store Energy, Low VDDmin, 8T2R Non-Volatile Latch and SRAM with Vertical-Stacked Resistive Memory (memristor) Devices for Low Power Mobile Applications,” *IEEE J. Solid-State Circuits*, vol. 47, pp. 1483-1496, Jun. 2012.

[10] M. Rasquinha et al., “An Energy Efficient Cache Design Using Spin Torque Transfer (STT) RAM,” in *Proc. ISLPED*, 2010.

[11] P. Zhou et al., ‘‘A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology,’’ in *Proc. ISCA*, 2009.

[12] P. Zhou et al., “Energy Reduction for STT-RAM using Early Write Termination,” in *Proc. ICCAD*, 2009, pp. 264-268.

[13] R. Ubal et al., “Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors,” in *Proc. SBAC-PAD*, 2007.

[14] A. Jaleel et al., “High Performance Cache Replacement using Re-Reference Interval Prediction (RRIP),” in *Proc. ISCA*, 2010.

[15] J. Wang et al., “OAP: An Obstruction-Aware Cache Management Policy for STT-RAM Last-Level Cache,” in *Proc. DATE*, 2013.

[16] A. P. Ferreira et al., “Increasing PCM Main Memory Lifetime,” in *Proc. DATE*, 2010, pp. 914-919.

[17] S.-Y Park et al, “CFLRU: A Replacement Algorithm for Flash Memory,” in *Proc. CASES*, 2006, pp. 234-241.