PROLONG: Priority based Write Bypassing Technique for Longer Lifetime in STT-RAM based LLC

Prabuddha Sinha*
prabuddha.19csz0008@iitrpr.ac.in
Indian Institute of Technology
Ropar, India

Krishna Pratik BV
bvkrishnap2001@gmail.com
RV College of Engineering
Bangalore, India

Shirshendu Das
shirshendu@cse.iith.ac.in
Indian Institute of Technology
Hyderabad, India

Venkata Kalyan Tavva
kalyantv@iitrpr.ac.in
Indian Institute of Technology
Ropar, India

Abstract

The rise of data-driven applications requires larger on-chip Last Level Caches (LLCs) in multicore systems which need denser chips with lower power consumption. Non-Volatile Memory (NVM) technologies such as Spin-Transfer Torque RAM (STT-RAM) based LLCs fulfill these requirements. STT-RAM based LLCs suffer from notable shortcomings, including higher write latency, increased write power consumption, and limited endurance. The primary cause of the low write endurance of an STT-RAM based LLC is the uneven distribution of write operations across the cache. To address the challenge of low write endurance in an STT-RAM based LLC while minimizing performance impact, we propose a unique inter-set wear leveling technique. It involves aggressively bypassing the write-back requests coming to the LLC from upper level of cache memories and writing directly to the main memory. Only those bypassed writes, which may be accessed again in the future, are stored in an SRAM buffer. Bypassing writes for a specific block in the LLC to SRAM Buffer or Main Memory is contingent upon satisfying certain conditions based on the set-wise write count and the priority of the block. Through extensive computational analysis, we have determined that the application of this technique in a quadcore setup reduces inter-set write variation by 84% and intra-set write variation by 19%. A lifetime improvement of 35 times as compared to the baseline STT-RAM based LLC can be achieved. Importantly, our approach maintains the system performance.

CCS Concepts

• **Computer systems organization → System on a chip**; • **Hardware → Spintronics and magnetic technologies**; **Memory and dense storage**.

*Corresponding Author.

Keywords

STT-RAM, LLC, write bypassing, SRAM Buffer, Inter-set Write Variation, Lifetime Improvement.

ACM Reference Format:
Prabuddha Sinha, Krishna Pratik BV, Shirshendu Das, and Venkata Kalyan Tavva. 2024. PROLONG: Priority based Write Bypassing Technique for Longer Lifetime in STT-RAM based LLC. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (MEMSYS '24).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

With the onset of the Artificial Intelligence and Machine Learning era, the upcoming workloads will always be highly data intensive. Supporting such workloads require larger on-chip caches in order to minimize the off chip data movement that incurs extra latency. CPU architectures have been revolutionized over the years, but the traditional cache architectures are not able to scale up as such, creating a bottleneck in performance. Therefore, there is a need for a larger sized Last Level Cache (LLC). SRAM cells are commonly used for on-chip cache designs in today's multiprocessors. They have a few drawbacks such as requiring large area (i.e., relatively low density) and high leakage power. In view of these demerits various emerging Non-Volatile Memory (NVM) technologies are being explored as viable alternatives to the standard SRAM's. Phase Change Random Access Memory (PCRAM) [10], Resistive Random Access Memory (RRAM) [13] and Spin Transfer Torque Random Access Memory (STT-RAM) [8] have been proposed as NVM's. NVM technologies offer several advantages over traditional SRAM technology including higher density, non-volatility, improved scalability and low leakage power. Nonetheless, there are a few major drawbacks of using NVM's as on-chip caches mainly the expensive write operations. NVM's exhibit high write power consumption, low write endurance and high write latency. The long write latency of the NVM's place undue pressure on the LLC request queue and can lead to congestion[14]. The write endurance is also a concerning factor with RRAM [34] supporting only 10^{11} write operations while PCRAM[24] supporting upto 10^8 write operations. Whereas, STT-RAM [22] is expected to sustain 10^{15} write operations but after experimentation the actual value is less than 4×10^{12}. Among the emerging NVM technologies, STT-RAM is considered as an adoptable alternative of SRAM for LLC design.

When applications are run on various cores of a multiprocessor simultaneously, because of their inherent behaviour, they then create write variation (WV) throughout the LLC that results in drastic drop in the lifetime of the STT-RAM based LLC. The write variations are categorized into two: *Inter-set* write variation and *Intra-set* write variation[1]. Uneven distribution of writes between cache sets is known as inter-set write variation. This imbalance means that certain sets within the cache experience significantly higher writes than others, leading to rise in the failure rate in the heavily written sets compared to their counterparts. On the other hand, intra-set write variation arises from differing write frequencies among individual blocks (way/coloumn) within a cache set. Specifically, certain blocks within the set endure a disproportionately high number of writes compared to others, resulting in the accelerated deterioration of these heavily utilized blocks. These fluctuations in write behavior not only shorten the lifespan of the STT-RAM LLC but also contribute to a reduction in LLC capacity over time. The lifetime of the STT-RAM LLC is extended with a decrease in write variation both within and between sets. The quantity of writes is undoubtedly another significant factor that has a direct impact on the lifetime of STT-RAM LLC.

Numerous wear-levelling techniques have been proposed over the years, including both inter-set wear-levelling and intra-set wear-levelling techniques. Wang et al. [27] presented i^{2}WAP a combination of both PoLF and SwS. PoLF (Probabilistic Set Line Flush) is a write blocking method to a specific cache block in the set. It marks a block as invalid after a fixed number of writes, which is determined by a Flush Threshold (FT). SwS (Swap Shift) is employed to mitigate inter-set write variation, which entails modifying the mapping of two sets following an epoch, only if a certain threshold is reached, which is referred to as the Swap Threshold (ST). Jokar et al. [9] introduced Sequoia that comprises of G-OAS and WAD. G-OAS (Grouped On-Access Interset Swapping) proposes a set remapping technique where the mapping of write hot sets are swapped with those sets that are not write hot. WAD (Write-back aware intra-set displacement) checks for hot cache lines, making it a candidate for displacement. It follows a two-phase invalidation-displacement policy, first invalidating the hot line while protecting its metadata, then displacing it if dirty. Agarwal et al. [2] proposed three different approaches window based approaches namely Static Window Write Restriction (SWWR): where the window size is static, Dynamic Window Write Restriction (DWWR): where window size changes dynamically and Dynamic Way Aware Write Restriction (DWAWR).

Figure 1 and Figure 2 illustrates the reduction in write variations (WV) and the Normalized IPC reduction respectively for various state-of-the-art techniques as compared to the baseline. Here, the baseline is considered as STT-RAM LLC with LRU replacement policy and no wear levelling technique. The specifics about the experimental setup is discussed in Section 4. From the figures, it is evident that techniques such as SWWR, DWAWR, PoLF, and WAD are unable to decrease the inter-set WV to the same extent as intra-set, while techniques such as SwS and G-OAS is able to decrease only the inter-set WV. Furthermore, the majority of these

[1]Throughout the paper we assume three levels of cache memories where the LLC (L3) is designed with STT-RAM. All the cache memories are considered as set-associative cache with 64B block size.

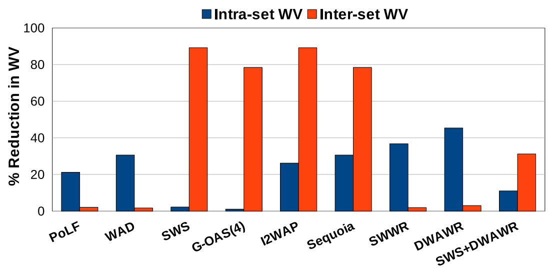

Figure 1: Percentage Reduction in Write variations (WV) in the existing wear-levelling techniques over baseline (LRU)

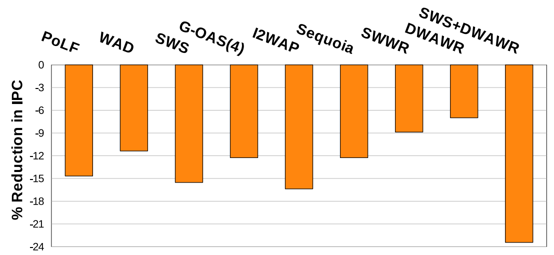

Figure 2: Percentage Reduction in IPC in the existing wear-levelling techniques over baseline (LRU)

existing techniques experience a drop in performance compared to the baseline. Figure 2 illustrates that there is a decline in their performance ranging from 7% to 23%. We combine the best of both the best techniques: SwS + DWAWR and have discovered that it did not perform as expected. The execution of SwS and DWAWR are not conductive as they interfere in each others smooth functioning. The reduction in both inter-set and intra-set write variation was significantly lesser compared to when the techniques were applied individually. The IPC was also reduced significantly. SwS always leads to additional remapping. The blocking of writes into ways due to DWAWR can cause further increase in the remapping and migration, which thereby increases the write count of the LLC. The existing wear-leveling techniques are insufficient in effectively minimizing inter-set write variation as well as intra-set write variation of STT-RAM LLC. These techniques can enhance the lifetime of a STT-RAM LLC by maximum of up-to 9 times. To this end we propose PROLONG.

In this work, we have introduced a highly effective wear-leveling technique PROLONG, that greatly minimises both the inter-set and intra-set WV and also reduces the write count. Here, we employ priority based write bypassing as an approach to minimize writes in the cache sets that experience a high write count. The primary focus of this work is to understand the rationale behind the intensity of bypassing and the selection of cache sets for bypassing. The cache set selection helps in redistributing the writes across the whole cache. An in-depth discussion is provided in Section 3. The primary contribution of the work can be classified as:

- We observe that PROLONG minimizes the variation in write operations within a STT-RAM LLC through write bypassing.

Experimentally it is observed that there is a decrease in inter-set WV of up-to 83% and intra-set WV up-to 19.5% w.r.t. baseline.
- STT-RAM LLC experiences a substantial enhancement in its lifetime up-to 35*x*.
- PROLONG preserves the enhancement in lifetime while maintaining system performance. It demonstrates a performance improvement up-to 2%.
- Rigorous experimental analysis thoroughly examines both the proposed concept and the existing concepts.

To the best of our understanding, no research has been conducted on utilizing the write bypassing concept to mitigate write variation (both inter-set and intra-set) in an STT-RAM LLC. The existing techniques for reducing write variation primarily emphasise on distributing the writes across the cache, rather than reduction in the write count. Our technique provides a method which will reduce both the write count in the STT-RAM LLC and also the write variation.

The rest of the paper is organised as follows. The background and motivational discussions are given in Section 2. Section 2.3 discusses the mathematical relationship among write-variations, number of writes and lifetime. The proposed idea is discussed in Section 3. The experimental setup and the results analysis are discussed in Section 4 and Section 5 respectively. Related works are discussed in Section 6. Finally the paper concludes in Section 7.

2 Background and Motivation

This section highlights the characteristics of STT-RAM, the NVM that we have used to design the LLC. It further discusses the different types of write variations plaguing STT-RAM and the motivation behind our architectural changes.

2.1 STT-RAM Cell

Figure 3 (a) shows the conceptual layout of an STT-RAM cell. This cell comprises of an access transistor alongside a Magnetic Tunnel Junction (MTJ). The MTJ structure consists of a tunnel barrier, composed of MgO, sandwiched between two ferromagnetic layers. The reference layer, possesses a fixed magnetization direction. Conversely, the magnetization direction of the other layer, termed the free layer, is variable and can be altered by a spin-polarized current. The tunnel barrier serves as a thin insulating layer separating the two ferromagnetic layers. These layers magnetization directions encode the stored data bit within the cell. Specifically, when the magnetization directions are anti-parallel, the resulting high resistance represents a logical '1' as shown in Figure 3 (c), whereas parallel magnetization denotes a logical '0' with low resistance as shown in Figure 3 (b).

Both reading and writing operations are performed by creating a voltage differential between the source and a bit-line. To write a '0' in the STT cell, a significant positive voltage is applied between the source and a bit-line, while writing a '1' involves applying a substantial negative voltage. Conversely, the read operation involves applying a minor voltage between the source and a bit-line, generating a current that is subsequently compared with a reference current to determine the '0' or '1' cell state.

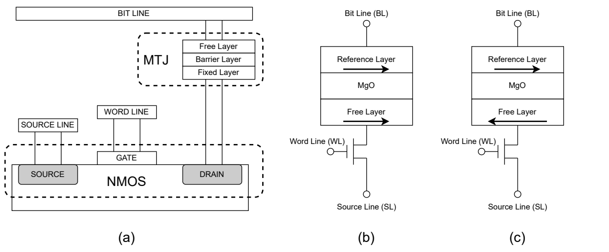

Figure 3: Layout of an STT-RAM cell

2.2 STT-RAM based LLC (STT-RAM LLC)

STT-RAM offers a wide range of advantages over conventional SRAM based LLC and other NVM's too [30], [22]. The general characteristics of an STT-RAM cell as compared to a SRAM cell is shown in the Table 1. Here F depicts the smallest feature size within a specific technology node.

Table 1: Comparison between SRAM and STT-RAM cell.

Parameters	SRAM	STT-RAM
Cell Size(F^2)	120-200	6-50
Write Endurance	10^{16}	4×10^{12}
Speed(Read/Write)	Very Fast	Very Fast/Slow
Leakage Power	High	Low
Dynamic Energy(R/W)	Low	Low/High
Retention Period	Very Low	High

Therefore, even though STT-RAM has a lot of advantages over SRAM like smaller cell size, low leakage power and higher retention time as depicted in Table 1, it comes with its own set of disadvantages. The primary obstacle facing on chip memory based on STT-RAM LLCs, is the elevated write latency and write energy consumption in comparison to SRAM. Although various techniques and hybrid technologies as discussed in Section 6 have been proposed in order to reduce the write latency of the STT-RAM LLC none have been successful to do both, i.e. improve performance and extend the lifetime of the STT-RAM LLC. Our work focuses on providing effective solutions to both these problems.

2.3 Write Variation and Lifetime

Write variation poses a substantial challenge in the design of cache and write limited memory technologies. Extensive write variation can severely diminish the cache's lifespan. A mere fraction of heavily written memory cells can disable an entire cache or memory system, despite most cells not nearing wear-out. Implementing wear-leveling in caches introduces additional challenges due to variations in write counts both within individual cache sets (intra-set variations) and across different cache sets (inter-set variations). Our work aims at reducing the write variation in the STT-RAM LLC. For the purpose of quantifying the write variation, we employ coefficients. Inter-set write variation and intra-set write variation are represented by Equation 1 and Equation 2 respectively as defined in [27].

$$InterV = \frac{1}{W_{avg}} \sqrt{\frac{\sum_{i=1}^{N} (\sum_{j=1}^{M} w_(i,j)/M - W_{avg})^2}{N-1}} \quad (1)$$

$$IntraV = \frac{1}{W_{avg}.N} \sum_{i=1}^{N} \sqrt{\frac{\sum_{j=1}^{M} (w_{i,j} - \sum_{j=1}^{M} w_{i,j}/M)^2}{M-1}} \quad (2)$$

Here, $w_{i,j}$ is the block write count of ith set and jth way. Average write count W_{avg} is defined as :

$$W_{avg} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} w_{i,j}}{NM} \quad (3)$$

M is the total number of cache ways in a set and N is the total number of cache sets. Cache wear-leveling aims to mitigate both inter-set and intra-set variations while also limiting the maximum write count. Therefore, configurations having lower write variation values shows better improvements in the lifetime as the maximum write counts are lower.

Relative lifetime is used to quantify the lifetime of the cache. It can be defined as the inverse of the maximum write count on a cache line in the LLC [2]. It is a derivative of raw error tolerant lifetime which is the time taken by the cache for the first error to occur in a cache line after specific number of writes to that cache line. The term "raw" implies that no error recovery mechanism is considered.

2.4 Motivation

This study focuses on minimizing inter-set write variation while maintaining the performance of the STT-RAM LLC. A key hurdle in addressing inter-set write variation lies in the fixed mapping policies inherent in set-associative caches. In such caches, a block is assigned to a specific set, and relocating a block from one set to another necessitates a modification in its block-mapping policy. This can lead to a drop in IPC of the system. Consequently, strategies to alleviate inter-set write variation while reducing the effects of remapping policies have remained largely unexplored for STT-RAM LLC's. This prompted us to introduce a proficient inter-set wear leveling technique.

Through analysis of the effect of write variation, we performed experiments utilizing a simulator for a quad-core system equipped with a 16 MB 16-way set-associative STT-RAM LLC lacking wear leveling support with standard LRU replacement policy. We have used *Mix 2* shown in Table 7. The configuration is executed for 250 Million instructions. Figure 4 shows us the non uniform write distribution across cache sets that are prevalent. It depicts the write count after every 100 sets. Nonuniform writes to an STT-RAM LLC is the major reason for decreasing endurance as well as the lifetime of the LLC, thereby inducing errors. The lifetime of the LLC is directly dependent on the maximum write count. The maximum write count among all the sets is 3516 while the write average is only 564. This clearly shows the huge disparity in write counts that is present among the sets. If the inter-set write variation is removed then the overall lifetime will be enhanced by $6.23x$ in this case.

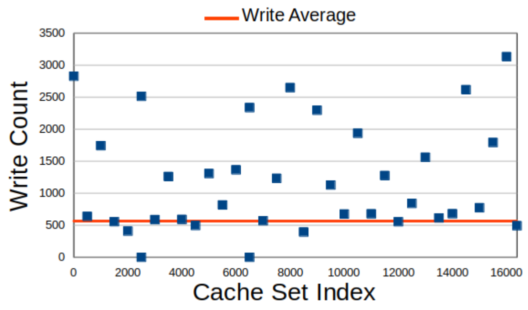

Figure 4: Non-uniform write distribution across cache sets.

3 PROLONG: The Proposed Work

3.1 Proposed Architecture

In order to support PROLONG we propose a few architectural changes as compared to the traditional memory hierarchy. We introduce three new components namely writes set counter (WSC), a small SRAM buffer located between the STT-RAM LLC and the Main Memory and a Liveness Score Counter (LSC). Figure 5 depicts the newly proposed architecture. RQ depicts the Read Queue and WQ depicts the Write Queue of the LLC. The SRAM Buffer is placed in between the LLC and MM in the memory hierarchy and before checking for any block in the MM we check for it in the SRAM Buffer. Each block present in the LLC is also equipped with the "Liveness Score" counter (LSC) while each and every set is equipped with a WSC.

3.1.1 Writes set counter (WSC). Each set of the STT-RAM LLC is equipped with a dedicated 8-bit write counter. These write counters are updated whenever there is a write into that specific set of the STT-RAM LLC. Whenever the write counter is saturated in a specific epoch, for the next epoch the write count for that specific write counter is halved. This way any new incoming writes can again be counted in that counter. This way the significance of the write intensity in that counter is not explicitly dismissed for the upcoming epochs.

3.1.2 SRAM Buffer. A small SRAM buffer is placed in between the STT-RAM LLC and the Main Memory. This buffer is used to store selective high priority writes from write hot ways, that have been bypassed from the STT-RAM LLC by PROLONG to reduce the inter-set write variation. The writes that are stored in the SRAM buffer are those writes that have high "liveness score", i.e. these writes have a high probability to be recalled and may be used again in the near future by the LLC. We conduct studies with buffer size varying from 32KB to 512KB. The buffer is divided into multiple parts of 32 entries and a block can be mapped into a fixed part like set-associative cache. The FIFO replacement policy is used for each part separately. Prior to sending any LLC miss request to the main memory, the block is first searched in the buffer. In case of a hit in the buffer, the block is moved back into the LLC. The main purpose of this buffer is to reduce the side effects of bypassing in PROLONG and maintain the performance of the system as time taken to service a block from memory will be drastically greater than the time taken to service it from the small SRAM buffer in case of a hit.

3.1.3 Liveness Score Counter (LSC). Every block in the STT-RAM LLC is equipped with a 2-bit "liveness score" counter (LSC) whose value is updated upon every access to the blocks from L2. It is used to store the liveness score of the block.

Liveness Score of a block is defined as the reuse probability of the LLC block in the L2 cache. 32 observer sets present in both the L2 and the LLC are used in order for learning the reuse probability of the block similar to [11], [14]. It uses program counters (PC's) to sample the blocks. At the L2 cache, hashed PC's (Instruction addresses that last accessed the cache line in the L2) is maintained as a reference table. This table consists of four 10-bit counters each mapping to a liveness buckets ranging from 0 to 20%, 21 to 50%, 51 to 70%, and 71 to 100%. The proportion of writes in an access PC being recalled from the LLC is defined as liveness. 20% liveness implies there were 20 reads among 100 writes to the LLC in a single PC based access. Liveness counters are decremented during eviction of a block from an L2 observer set. The sampler technique described in [11], [14] provides us a reference for usage of partial-PC tags for blocks in the observer sets. Initially whenever a block is written into L2 cache the liveness counters are set as 0. Upon a LLC hit, the liveness counters of the access PC are incremented. Eviction and hits have different changes in the liveness counters for observer sets. Eviction of a block having 21-50% liveness counter leads to counter decrement of 2 and on a hit it is incremented by 10 for both 21-50% counter and 0-20% counter. A positive value of this counter indicates that it has at least 20% liveness. A 2-bit liveness score value is also assigned to every new incoming block in the L2 if it is not in the observer set, based on its previous PC based observer set value. The highest liveness bucket has the highest priority. For example, if both 71 to 100% and 51 to 70% liveness counters are both positive, the line is assigned the 71 to 100% live score. Only four buckets are being tracked for each block, hence 2 bits are required for the liveness score in L2. The liveness score is part of the L2 evicted block. An LSC is assigned to store this liveness score value in LLC. Initially, any new write will have a default liveness score of 0 corresponding to the 0-20% liveness bucket.

3.2 Working of PROLONG

The main idea is to selectively enable bypass for only the heavily written sets. This helps in making the writes uniform throughout the cache sets. The write bypass depends on two parameters: the liveness score counter (LSC) and the write set counter (WSC).

The LSC of any block in the LLC can range from 0 to 3. Table 2 represents how the write liveness score buckets and their status bits are assigned based on their liveness percentage.

Write Hot (WH) bucket selection among sets: Whenever there is an incoming write, that can either be a L2 write-back or a Last Level Cache miss, the write set counter is incremented. The WSC indicates the write hotness of the set. A cache-set is set to be write hot if it has garnered repeated writes to it within an epoch. After every epoch, which is a certain number of instructions, we identify the write hot sets and then we target priority based write bypassing from these write hot sets. These write hot sets are further classified into three categories equally and different degrees of bypassing are applied to each category. The number of sets selected to be bypassed is based on the bypass aggressiveness percentage. We consider bypass aggressiveness from 2% heavily written sets to 15% heavily written sets. The chosen hot sets are subsequently distributed evenly among three buckets, determined by their respective write counters. Sets with the highest write count are allocated to bucket H_3, those with a medium write count are assigned to bucket H_2, and sets with the lowest write hotness among the hot sets are placed in bucket H_1. Remaining sets that show less write activity, are allocated to H_0 bucket. To store the bucket status bit for each block, 2 bits per cache-set are required. Table 3 shows how the write hot buckets and their status bits are assigned.

Table 2: Bucket Distribution among sets based on Liveness Score (LS).

LSC Bucket	LSC Bit Value	Liveness Percentage
LS0	00	0-20%
LS1	01	21-50%
LS2	10	51-70%
LS3	11	71-100%

Table 3: Bucket Distribution among sets based on Write Counter.

WH Bucket	Bit Value	Set Status
H_0	00	Sets that are not write hot
H_1	01	Write hot sets with low writes
H_2	10	Write hot sets with medium writes
H_3	11	Write hot sets with high writes

Bypassing decision and aggressiveness: At the beginning of each epoch, adjustments to the write counters and write hot buckets for every set are necessary. Write hot sets among all sets in an LLC are identified based on the level of bypassing aggressiveness. The more aggressive the bypassing, the greater the number of write hot sets selected. These write hot sets are divided into write hot buckets and the bucket values for each set are assigned accordingly. The distribution of write hot sets is allocated uniformly among the three write hot buckets: H_1, H_2 and H_3 as depicted in Table 3.

Bypass Aggressiveness is defined as percentage of sets considered as hot sets. Suppose if the write bypassing aggressiveness is 15%, we consider 15% of all sets in the cache as write hot. The top 5% write hot sets are assigned to H3 then the medium 5% is assigned to H2 and the lowest 5% among the hot sets is assigned to H1. The LSC of the blocks is automatically re-calibrated after every cache access. When a write-back request is generated from L2, the block needs to be either written in LLC or bypassed. The decision is taken based on the liveness score of the block and the WH bucket it maps to. It first inspects the WH bucket of the set to which the write block needs to be written. If it is in H_0, there will be no write bypassing. Conversely, if the writes are identified to be directed to the write hot sets (H_3, H_2 or H_1), we need to assess whether the incoming writes should be bypassed or not. Write Bypassing logic for incoming write-backs to the STT-RAM LLC sets are depicted through Table 4.

Table 4: Write Bypassing based on Write Hot Sets (WH) and LSC Buckets.

	WH Bucket	LS Bucket	Bypassing
1	H_0	LS0, LS1, LS2, LS3	No Bypassing
2	H_1	LS3, LS2, LS1	No Bypassing
3	H_1	LS0	Bypass to Main Memory
4	H_2	LS3, LS2	No Bypassing
5	H_2	LS1	Bypass to Buffer
6	H_2	LS0	Bypass to Main Memory
7	H_3	LS3	No Bypassing
8	H_3	LS2, LS1	Bypass to Buffer
9	H_3	LS0	Bypass to Main Memory

Table 4 indicates that selection 1, 2, 4 and 7 will always lead to no write bypassing from the designated set, i.e. the write will occur in its original state. This shows that blocks with higher liveness score, specifically LS3 will not be bypassed in any situation. Selection 3, 6 and 9 indicate that low priority blocks with LS0 are not needed to be stored in write hot sets and are directly bypassed to the main memory. The remaining selection 5 and 8 depict the conditions necessary in order to bypass writes from the write hot sets to the SRAM buffer.

3.3 PROLONG Algorithm

The work flow of the proposed PROLONG is depicted through the Figure 5. It comprises of the following steps:

Step-1: Upon receiving a write-back request from L2, it is inserted into the write queue (WQ) of the LLC, and its LSC value is evaluated from the incoming LS value.

Step-2: The LSC bucket value of the write request and the WH bucket value of the set is both compared and the decision of bypassing is taken based on the logic depicted in Table 4.

Step-3: If both conditions are met simultaneously, i.e., the LS bucket and the WH bucket are favourable than, the write may be bypassed to the SRAM buffer if they have a high LS value.

Step-4: Writes with low LS that have been selected for bypass are directly sent to the main memory.

These processes are calculated simultaneously with the working of the cache and are not under the critical path. However, the the SRAM buffer access time during a LLC miss comes under the critical path. We have considered this buffer access time in our experimental analysis (Section 5.3).

4 Experimental Setup

4.1 Simulator Setup

For all the experimental implementation of this work we use the ChampSim Simulator [6]. ChampSim replicates a diverse multicore system featuring various cores and a custom memory hierarchy, allowing each out-of-order core to be individually configured according to specific requirements. Our experiments are conducted

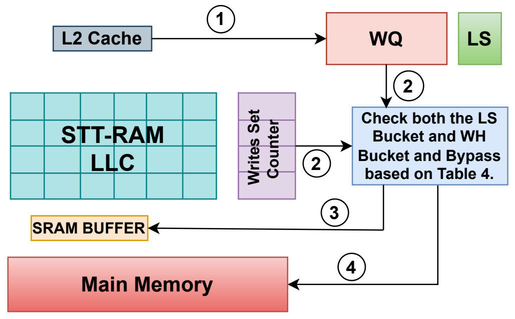

Figure 5: Architecture and Steps in PROLONG Algorithm

Table 5: Simulation parameters of the baseline system.

System Components	Parameters
Core	Out-of-order, bimodal branch predictor, 4 GHz with 6-issue width, 4-retire width, 352-entry ROB
L1I	32 KB, 8-way, 4 cycles
L1D	48 KB, 12-way, 5 cycles
L2	1MB, 8-way, 10 cycles, LRU
LLC	4MB, STT-RAM, 16-way, WL: 100 cycles, RL: 20 cycles, LRU
MSHRs	8/16/32 at L1I/L1D/L2, 64/core at the LLC
DRAM controller	64-entry RQ and WQ, reads prioritized over writes, Burst write: 6/8th of queue size
DRAM chip	4KB row-buffer per bank, open page, burst length 16, t_{RP}: 12.5ns, t_{RCD}: 12.5ns, t_{CAS}: 12.5ns

on a single and multicore (mainly 4 core) system with three levels of cache hierarchy. L1 and L2 are SRAM based caches. The L3 (LLC) is designed with STT-RAM. For the multicore setup L3 is a shared cache. The ChampSim is modified to support the STT-RAM LLC. Table 5 shows us the detailed parameters of the system used. WL denotes Write Latency while RL denotes Read Latency. All our timing parametrers have been calculated with the help of CACTI [23].

We compare our work with existing approaches namely, PoLF [27], WAD [9],SwS [27], G-OAS [9], SWWR [2] and DWAWR [2] and the baseline STT-RAM LLC. The baseline LLC uses LRU replacement policy and has no wear-levelling policy implemented. In PoLF the flush threshold (FT) value has been assigned as 16 for the simulations. WAD uses the clean-LRU block displacement algorithm along with a 3-bit saturating counters (SC). SwS only helps in swapping the mapping of two sets at once and all other sets are shifted gradually. Swapping only occurs if the total write count of the cache crosses a certain swap threshold (ST). G-OAS groups the cache sets into small groups (4,8,16,etc.) and swaps the write hot sets with the cold sets within the different groups. G-OAS(4) indicates that 4 groups of sets are considered based on their write

Table 6: Different PROLONG configurations used.

Bypassing Aggressiveness %	Buffer Sizes
PRO(2%)	32KB, 64KB, 128KB, 256KB and 512KB
PRO(5%)	32KB, 64KB, 128KB, 256KB and 512KB
PRO(10%)	32KB, 64KB, 128KB, 256KB and 512KB
PRO(15%)	32KB, 64KB, 128KB, 256KB and 512KB

hotness similar to our configuration. SWWR uses a window size of 4 ways with a 12-bit write counter per window. DWAWR uses a 11-bit write counter per way for counting the writes. Different configurations of our work are shown in Table 6. We implement PROLONG with varying configurations based on bypass aggressiveness and SRAM buffer size. PRO(x%) indicates that the bypass aggressiveness of PROLONG is x%. Our work uses only an 8-bit counter per set for WSC because it is enough in calculating the writes over our chosen epoch size. An extra 2-bit counter is used for every set along with the WSC which is used to store the WH bucket value. For our experiments, we consider epoch size to be 10^5 instructions. In a single-core setup for each workload, the simulation commences with a 100 million instruction warm-up phase, followed by an additional 1 billion instruction run to reach completion. In case of multicore setup a 50 million instruction warm-up and a 250M instruction run was deployed for our quad core systems.

4.2 Workloads

We have used write intensive benchmarks from SPEC CPU 2006 benchmark suites [7]. We first simulated the SPEC CPU 2006 workloads and identified those workloads that are write intensive in LLC. A few write intensive graph benchmarks from the GAP benchmark suite[4] were also used for our simulations. Table 7 identifies all the write intensive workloads on which our single-core simulations have been run.

Table 7: Workloads for single-core system.

Benchmark Suite	Write Intensive Workloads
SPEC 2006	cactusADM, gcc, GemsFDTD, lbm, leslie3d, libquantum, mcf, milc, soplex, sphinx3, wrf, xalancbmk, zeusmp
GAP	bc, bfs, cc, pr, sssp

For the quad-core setup, we use 4 different workload mixes. Each mix has different characteristics. The workload mixes are depicted in Table 8. 'L' denotes low write intensive workloads and 'H' denotes high write intensive workloads. 'G' denotes Graph workloads.

5 Results Analysis

5.1 Single-core analysis

In this section we compare different wear-levelling techniques with the baseline STT-RAM LLC considering various metrics. Among the different configurations of PROLONG, we consider the configurations only with 512KB buffer in this analysis with varying

Table 8: Workloads for Quad-Core System.

Mix Type	Workloads
Mix1-LLHH	gobmk, gromacs, mcf, libquantum
Mix2-HHHH	mcf, libquantum, lbm, xalancbmk
Mix3-LLLL	gobmk, gromacs, gamess, namd
Mix4-GGGG	pr (page rank), bc (betweenness centrality), bfs (breadth first search), sssp (single source shortest path)

degrees of aggressiveness. A detailed sensitivity analysis with other PROLONG configurations is given in Section 5.3. Figure 6 shows the reduction in inter-set write variation of different wear-levelling techniques, normalised to baseline. It can be observed from the figure that, on an average the inter-set write variation reduces by around 84% in the case of PROLONG (15%) configuration. Among the existing techniques, SWS and G-OAS(4) achieve inter-set write variation reduction of 89% and 78%, respectively. Remaining techniques perform poorly. The main reason for the reduction in inter-set write variation in the case of PROLONG (10% & 15%) is aggressive bypassing of writes from the heavily written sets. However, it has been observed that excessive aggressive bypassing is not required as even the 2% aggressiveness achieves 74% reduction in inter-set write variation over the baseline. The higher the aggressiveness of bypass the greater the IPC of the system. More detailed analysis about aggressiveness is discussed in Section 5.3.

The reduction in intra-set write variation of PROLONG is comparable with the existing wear-leveling techniques. As shown in Figure 7, the reduction in intra-set variation for PROLONG is just above 18% normalized to baseline. This is comparable with previous state of the art 15% for PoLF and 29% WAD which are specialized intra-set write reduction techniques. The reason behind such reduction in the existing techniques is that all these techniques try to distribute the writes among the different ways of the set. There are two main reasons for reducing the intra-set write variation by PROLONG. First, it has been observed that the intra-set write variation is high on the sets having more writes. Hence, bypassing writes from heavily written sets also helps in reducing write variation within the set. Second reason is based on an insight from PoLF [27]. The idea discussed in PoLF achieves significant improvement in intra-set write variation by just invalidating random cache blocks. Bypassing writes also means invalidating the existing block that belongs to an heavily written set of the LLC. Hence, combining these two important reasons, PROLONG also shows competitive reduction (around 18%) in intra-set write variation.

The lifetime of a STT-RAM based LLC is dependant on the maximum write count of a single cache line. Bypassing writes from heavily written sets not only reduces the total number writes performed in the STT-RAM LLC but also reducing the maximum write count. Hence, PROLONG performs less number of writes in the LLC as compared to the other techniques. The existing techniques sometimes even increase the number of writes because of invalidating important blocks from the LLC. These important blocks, the blocks that are likely to be reused in the future, need to be fetched from the main memory again, leading to more writes at the LLC.

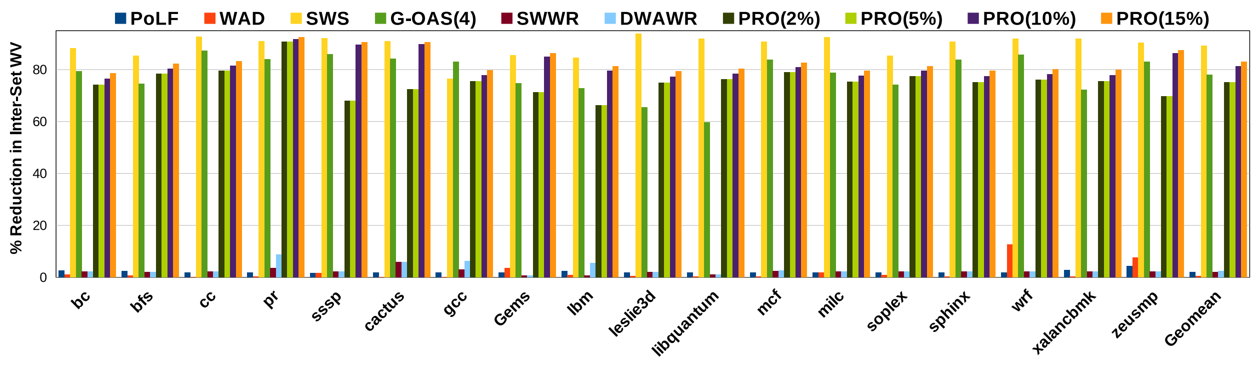

Figure 6: Percentage Reduction in Inter-set WV in Single-core System (higher the better).

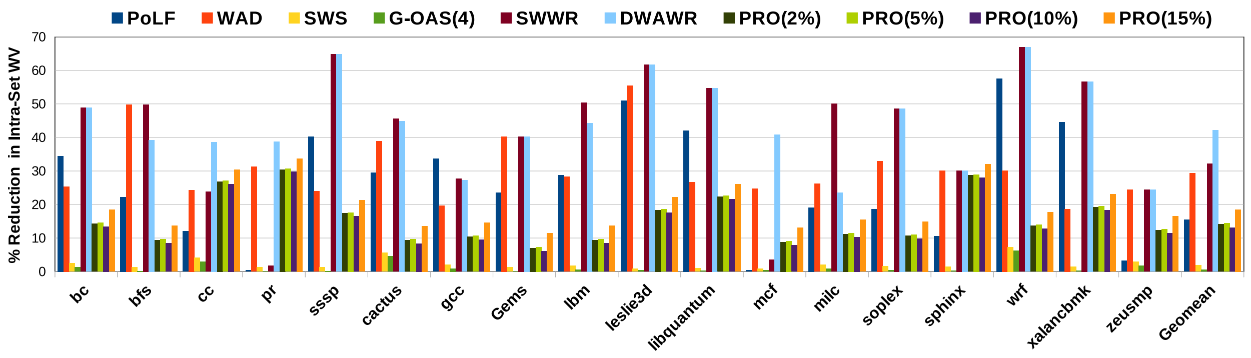

Figure 7: Percentage Reduction in Intra-set WV in Single-core System (higher the better).

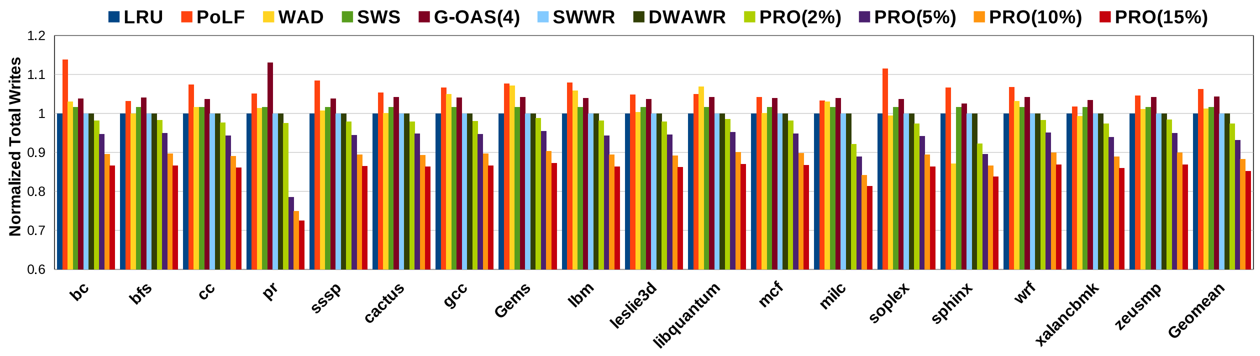

Figure 8: Comparison of Normalized Total Write Count in single-core Systems (lower is better).

Missed important blocks can potentially lead to performance degradation. However, in PROLONG, the write bypassing takes place based on a liveness score of the block. The liveness score indicates the possibility of the block being reused again in the near future. Therefore, the possibility of bypassing an important block is less as compared to the other state of the art. Figure 8 shows the comparison in number of writes by different techniques as compared to the baseline, i.e. Normalized Total Writes. It can be observed from the figure that the reduction in writes is more in PROLONG where in other techniques that do not employ write bypassing and the writes are either more or equal to that of the baseline (LRU). Figure 8 depicts the various configurations of PROLONG with varying degrees of aggressiveness (2-15%).

Figure 9 shows the improvements in lifetime over the baseline by the various wear-levelling techniques considered. As mentioned in Section 2.3, the lifetime depends on write variations and the maximum number of writes in a cache line. PROLONG improves all the three parameters as shown in Figure 6, 7 & 8. Therefore, Figure 9 shows significantly better lifetime than the other techniques. All the other existing techniques can enhance the lifetime at maximum

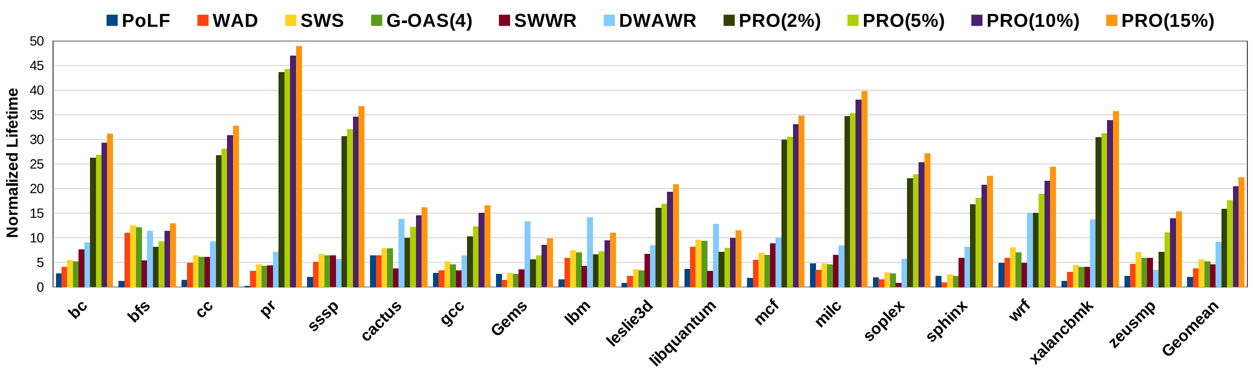

Figure 9: Comparison of Normalized Lifetime in single-core Systems (higher the better).

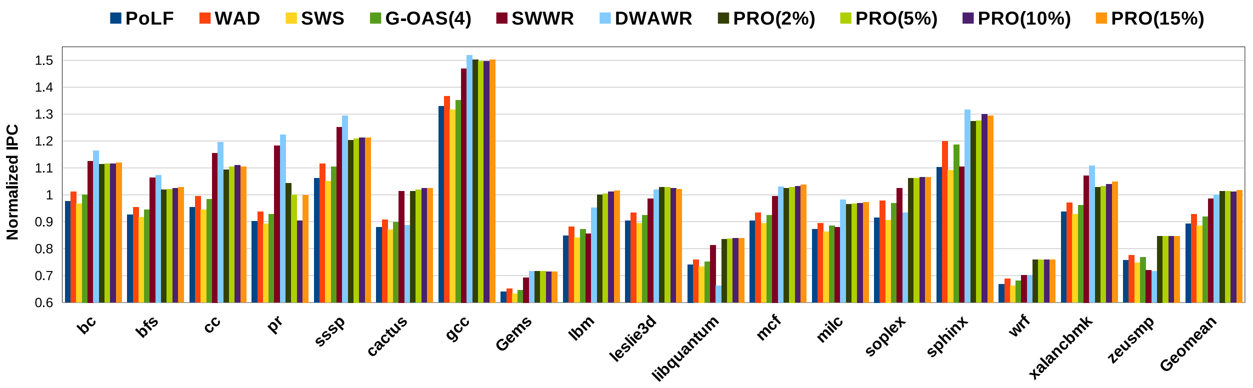

Figure 10: Comparison of Normalized IPC in single-core Systems (higher the better).

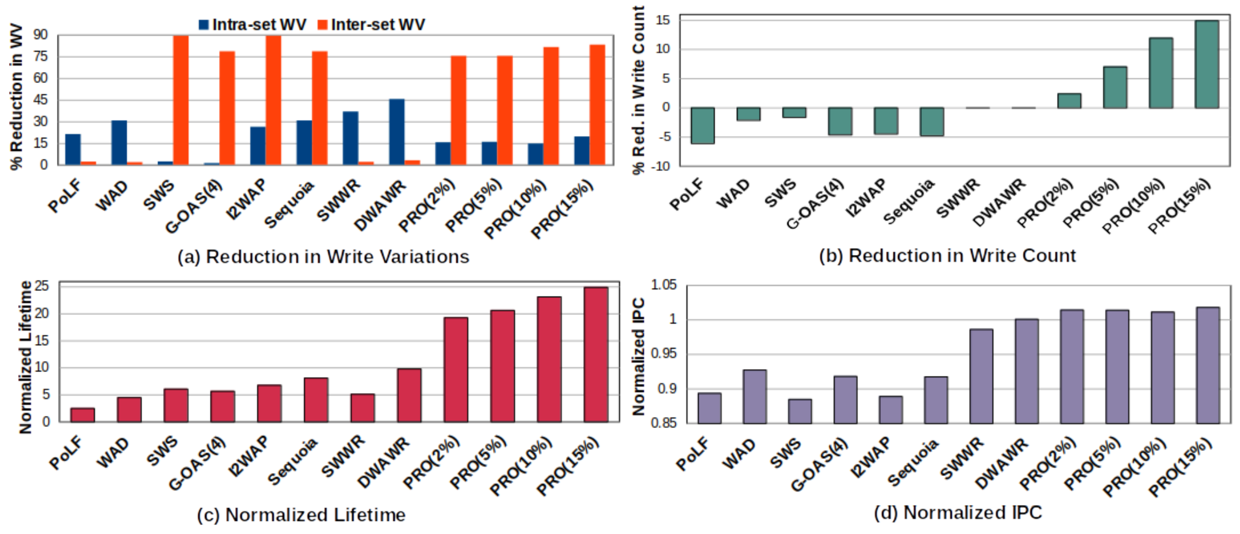

Figure 11: Average Comparison of PROLONG in Single-core Systems with State of the art.

by $9x$ wheres PROLONG enhances it by nearly $22x$, for the single-core configuration. Here, the lifetime is calculated as the normalized lifetime over baseline.

Impact on performance: While achieving significant improvement in lifetime, PROLONG does not degrade (on an average) the performance of the system. Normalized IPC is used as the metric to show the various performance improvements as compared to the other state of the art. Figure 10 shows that PROLONG matches the performance of baseline and in some cases namely, bc, cc, pr, sssp, gcc and sphinx, PROLONG achieves more than 10% performance improvement over the baseline. The average improvement ranges from 0-2%. The reason behind the improvement is because of bypassing the less important blocks from the cache. As discussed in Density [14], bypassing less important blocks reduces the congestion in the read/write queue at the LLC and hence improves the efficiency of the LLC. Because of lesser aggressive bypassing than discussed in Density [14], PROLONG does not attain performance improvements similar to Density. The other existing wear-leveling techniques, PoLF, WAD, SWS and G-OAS(4) show a significant drop in performance. Similar to PROLONG, SWWR and DWAWR almost achieve baseline performance. This is the main benefit of PROLONG as it improves the lifetime significantly as compared to the existing techniques without degrading the performance.

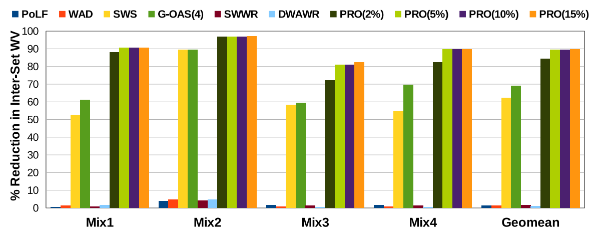

Figure 12: Percentage Reduction in Inter-set WV for Multicore Systems (higher the better).

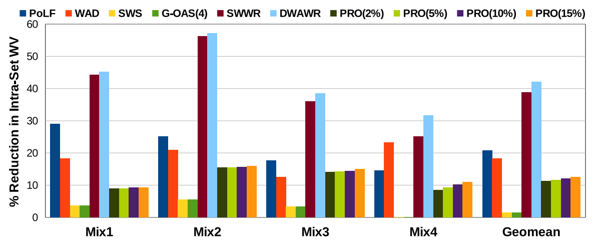

Figure 13: Percentage Reduction in Intra-set WV for Multicore Systems (higher the better).

Improvement analysis: Figure 11(a) shows the geomean reduction in terms of write variations both inter and intra-set write variation as compared to the baseline and other state of the art configurations. Figure 11(b) shows us the reduction in the total write count of PROLONG that occurs due to write bypassing. It is depicted as the percentage reduction in write count as compared to baseline. Other wear levelling techniques show an increase in the write count as compared to baseline. The number of writes may increase but we see that the maximum write count does not due to effective wear levelling. Figure 11(c) shows us how the lifetime is effected effected by the various existing techniques considered along with PROLONG. Figure 11(d) deals with the effect of the state of the art on the performance calculated as Normalized IPC, over baseline. The improvement shown in these figures are the geometric mean values of all the workloads considered for study. It can be observed that PROLONG reduces both inter-set and intra-set write variation by almost 84% and 19% respectively. The write count in existing techniques is more than the baseline wheres in PROLONG it reduces because of its efficient bypassing. As the bypass aggressiveness of PROLONG increases, the reduction in write count increases due to the bypassing of unimportant blocks. The significant improvement in write variations reduction and write count reduction makes PROLONG improve the lifetime as shown in Figure 11(d). An important point to observe here is that the higher the bypass aggressiveness, the higher the lifetime.

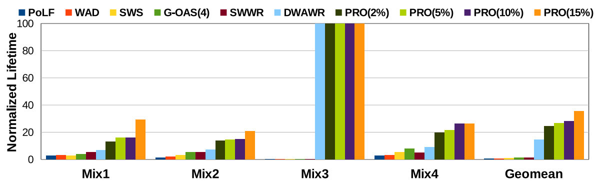

Figure 14: Normalized Lifetime w.r.t. baseline (LRU) for Multicore Systems (higher the better).

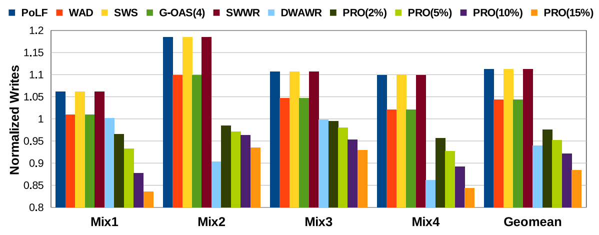

Figure 15: Normalized Total Writes w.r.t. baseline (LRU) for Multicore Systems (lower is better).

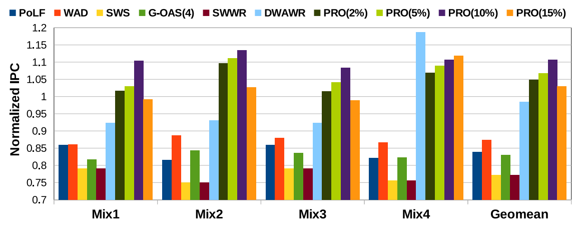

Figure 16: Normalized IPC w.r.t. baseline (LRU) for Multicore Systems (higher the better).

5.2 Multicore Analysis

Multicore experiments show the behaviour similar to single-core experiments but in an amplified scale. Figure 12 and 13 show the comparison of different wear-levelling techniques for reduction in inter-set and intra-set write variations. The reduction of intra-set variation by PROLONG is slightly less than single-core i.e. a maximum of 12%, while the inter-set write reduction is greater than the single-core system nearly by 89%. For multicore setup, Figure 14 shows that PROLONG enhances the lifetime up to 35x. However, a close observation shows that the enhancement is gradual and similar to single-core for *Mix2*. However, *Mix1*, *Mix3* and *Mix4* PROLONG shows a much higher enhancement. This occurs because of extensive bypassing of writes that will not be accessed again in the near future as these workloads have a major mix of read intensive instructions and graph workloads. The write count in the LLC decreases drastically with increasing aggressiveness of bypassing as depicted in Figure 15. Figure 16 shows the performance (Normalized IPC) improvements over the baseline. It can be observed from the figure that the performance of PROLONG is almost similar as baseline. However, the performance of the existing wear levelling techniques goes below the baseline.

5.3 Sensitivity Analysis

In this section we discuss about the various possible configurations of PROLONG and reason the optimum configuration. For the analysis of this section, PROLONG is considered with five different sizes of SRAM buffer: 32KB, 64KB, 128KB, 256KB, and 512KB. For all the earlier experiments we considered buffer size of 512KB. Each buffer is divided into multiple parts of 32 entries just like a set-associative cache such that the access time of the buffer can be reduced. Considering the latencies in Table 9, we perform experiments in this section. For each buffer size, PROLONG considered four categories of bypass aggressiveness: 2%, 5%, 10%, and 15%.

Table 9: The important hardware parameters of the SRAM buffer used in PROLONG. The parameters are extracted from CACTI [23].

Buffer Size	Access Latency	Static Power	Dynamic Energy per Access	Area Consumption
32KB	4.3ns	1.78nJ	19.44mW	6.1mm^2
64KB	4.4ns	1.81nJ	33.16mW	6.55mm^2
128KB	4.5ns	1.83nJ	60.74mW	7.4mm^2
256KB	4.8ns	1.86nJ	110.21mW	8.9mm^2
512KB	5.3ns	1.91nJ	168.65mW	10.45mm^2

Figure 17 shows the improvement in PROLONG over baseline in a single-core system. The improvements are shown for write variation, write count, lifetime and performance. It can be observed from Figure 17(a) that the improvements in intra-set write variations are almost similar with slight increments in reduction with increment in buffer size. The inter-set write variations slightly varies over different PROLONG configurations. However, the variations are within 6%. Figure 17(b) shows that the number of writes in LLC reduces with increase in bypass aggressiveness. A 512KB buffer with 15% bypass aggressiveness depicts a 14% decrease in the overall write count. The change in normalized lifetime (as shown in Figure 17(c)) over the different PROLONG configurations varies up to 1.5x. The minimal lifetime improvement is only 17x while the maximum is nearly 25x. Figure 17(d) shows that there is no significant improvement in the performance with an increase in the SRAM buffer size. The main reason behind this behaviour is the high write latency of the larger SRAM buffers. The write latency of the SRAM buffer increase slightly with its increasing size as depicted in Table 9.

Hence, from the above discussion it can be concluded that a PROLONG configuration with large SRAM buffer is not improving noticeably. Furthermore, they have a larger hardware overhead that make it an unattractive prospect. Buffer sizes that are small are also not optimal as it lags behind in the lifetime improvement. Therefore, buffer sizes that are in the middle like 128KB can provide the best of both worlds with minimal hardware overhead and high normalized lifetime for highly aggressive write bypassing is more advisable. The results from multicore setup also reflects the similar insights but in a more heightened sense.

5.4 Importance of SRAM buffer

In this section we discuss the importance of using an SRAM buffer in PROLONG. Since the bypass aggressiveness that we used for PROLONG is more than the aggressiveness used in Density [14], there is a likelihood that PROLONG bypasses important blocks. The buffer is used to store such blocks. Figure 18 shows the buffer hit per LLC bypass. It can be observed from the figure that the hit rate in buffer increases with increasing buffer size. This implies that the larger the buffer it stores more important blocks with high probability of being reused again. As the bypass aggressiveness increases, more loads are coming to the SRAM buffer and hence the hit rate is also increasing. However, from the figure, it can be observed that without the SRAM buffer there will be multiple request which need to be served from main memory. Hence, the SRAM buffer plays an important role from the performance perspective.

5.5 Comparison with other write bypassing techniques

Write bypassing was originally used for reducing the number of writes and enhancing the performance of STT-RAM LLC [14]. PROLONG uses it for enhancing the LLC lifetime by reducing the write variations and write counts. Figure 19 shows the connection between PROLONG and Density [14]. From the Figure 19 for single-core systems it can be observed that PROLONG enhances the lifetime by 22x where in case of Density the lifetime enhancement is almost negligible as compared to baseline. This is because, Density is designed to bypasses writes to reduce the congestion in read/write queue. It even bypasses less important blocks throughout the LLC. However, PROLONG does more aggressive bypass only from some selective sets. On the other hand, Density shows higher performance improvement over baseline as compared to PROLONG as depicted in Figure 20.

5.6 Hardware Overhead and Energy Consumption

Table 10 presents the total area overhead of PROLONG for a given buffer size. Overhead comprises of write counters, WH bucket counters, LSC apart from the buffer. Since the other hardware components in PROLONG are almost same as baseline, the additional energy consumed by PROLONG (AE_{pro}) can be calculated

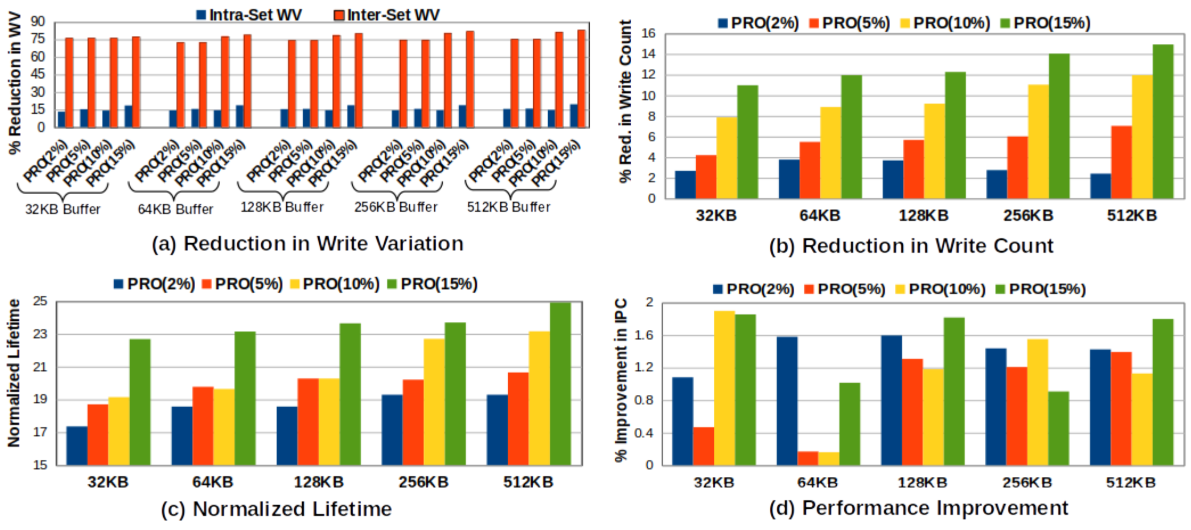

Figure 17: Comparison of different PROLONG configurations with baseline in a single-core environment. PRO(x%) means the bypass aggressiveness of PROLONG is x%. Each bypass aggressiveness value is experimented with 32KB, 64KB, 128KB, 256KB, and 512KB of buffer size.

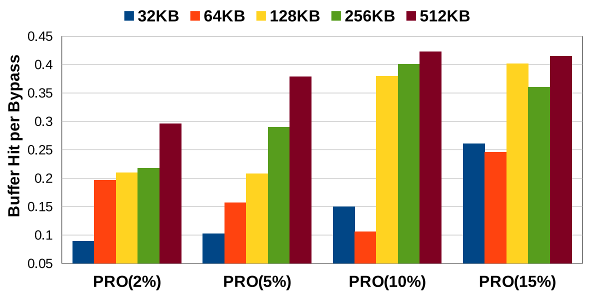

Figure 18: Buffer hit per write bypass in different configurations of PROLONG (higher the better).

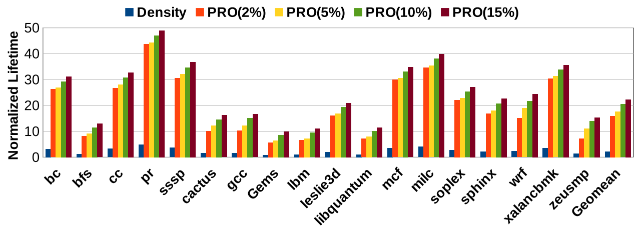

Figure 19: Comparison of Normalized Lifetime of PROLONG with Density [14] (higher the better).

as $AE_{pro} = E_{buff} - E_{write}$. Where E_{buff} is the energy consumed by the SRAM buffer and E_{write} is the energy saved by reducing the number of writes. We have used CACTI [23] to model the SRAM buffer. Experimental analysis with all sizes of SRAM buffer found that the value of AE_{pro} is always negative, implying that PROLONG

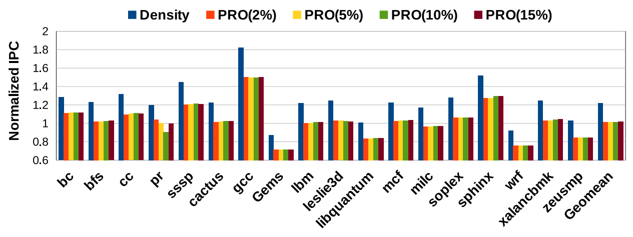

Figure 20: Comparison of Normalized IPC of PROLONG with Density [14] (higher the better).

do not have any additional energy overhead with any buffer size which is enough to get the benefit that we are expecting.

Table 10: Storage Overhead of Prolong.

Buffer Size	**Total Overhead**	**Overhead over Baseline**
32KB	53KB	1.29%
64KB	85KB	2.08%
128KB	149KB	3.63%
256KB	277KB	6.76%
512KB	533KB	13.01%

5.7 Lifetime Comparison Analysis

Table 11 represents the Normalized Lifetime of various state of the art techniques as compared with the different configurations of PROLONG over various write intensive workloads from both the SPEC-2006 benchmark suite and some write intensive graph workloads from the GAP benchmark suite. The lifetime values are normalized over baseline that is the STT-RAM based LLC employing LRU replacement policy with no wear levelling policy. Here, all the PROLONG configurations are set up with 512KB SRAM buffer. The graph benchmarks generally depict a higher degree of normalized lifetime up-to nearly 49x for PRO(15%). The highest normalized lifetime in case of SPEC 2006 workloads is only up-to 40x for PRO(15%). Table 11 shows that the geometric mean normalized lifetime for the state of the art is a maximum of 9x in case of DWAWR, whereas PRO(15%) shows the largest geometric mean normalized lifetime of 22.32x. A clear trend is visible in this Table 11 of increasing normalized lifetime with increasing degree of bypass aggressiveness.

6 Related Works

The existing literature presents a variety of proposed techniques for both inter-set and intra-set cache wear-leveling. We provide a brief overview of these policies. Additionally, we delve into existing inter-set cache wear-leveling techniques and a few proposed write bypassing techniques. Some hybrid caches were also introduced for the purpose of increasing the lifetime of the NVM based LLC's.

EqualChance, as reported by Mittal et al. [19], employs a strategy of transferring or swapping blocks within the cache set which are write-intensive, with invalid or clean blocks. The process is governed by a write counter threshold. However, this transfer or swap operation entails additional cycles and higher energy consumption, primarily due to the increased number of writes occurring within LLC. Mittal et al. [21] proposed LastingNVCache an intra-set write variation reduction technique that assigns a write counter for each block in the cache. When the counter value surpasses a predefined threshold, the new write on that block is bypassed. Mittal et al. [18] proposed a color mapping modification technique for reduction in inter-set write variation. Agarwal et al. [1] proposes a technique which divides sets into groups and a set in each group has both a normal part and a reserved part. Sets in a group are allowed to share the reserved part with those sets in the same group such that there is uniform distribution of writes. Wang et al. [26] proposed an Obstruction-Aware Cache Management Policy for STT-RAM Last-Level Caches (OAP). OAP focuses on periodic cache monitoring for the obstructive processes and then manages them accordingly.

A lot of hybrid cache techniques have been proposed over the years that comprises of both SRAM and NVM part. They aim to make the NVM part read intensive and the SRAM write intensive. Ayush proposed by Mittal et al. [20] is responsible for reduction in write variation because of block wise data movements among the different areas of the cache. Write migration happens to the SRAM part of the cache and read intensive blocks are stored in the NVM area. Lin et al. [17] proposed two access aware policies that were responsible to handle unbalanced STT-RAM wear-out and a partitioning scheme which is dynamic and depends on the wear-out. Wu et al. [31] highlights the use of different memory technologies at different cache levels and at the same level of cache with the help of which they have been able to achieve a bit of performance improvement but a high level of power reduction. HALLS [15] is a highly adaptable system that configures the best type of LLC based on the type of applications that are needed to be executed. Li et al. [16] proposed a Dual Associative Hybrid Cache (DAHYC) and their respective replacement policies which targeted having SRAM blocks along with NVM blocks in the same set of a cache. The replacement policy helps in managing the placement of the incoming write blocks in the volatile SRAM part or the NVM part efficiently. Wu et al. [32] proposed two different techniques where the on-chip cache hierarchy were made up of different type of memory technologies in different levels of the cache, i.e. Level based Hybrid Cache Architecture (LHCA) and also in the same level of the cache, i.e. Region based Hybrid Cache Architecture (RHCA).

There has been various other techniques that aim at improving lifetime. Wang et al. [28] proposed a unique solution where the cache has been partitioned into two and to balance the writes into those partitions they swap the two partitions based on the type of applications dynamically. The read intensive blocks will be positioned in the STT-RAM portion while the write intensive blocks will be positioned in the SRAM portion. They had proposed hardware based partitioning and software based partitioning as well. Saraf et al. [25] proposed a refresh aware replacement policy for write optimized STT-RAM's. They targeted those blocks to replace that were about to be invalidated instead of the recently refreshed block.

Write Bypassing techniques have been used in STT-RAM LLC in order to increase the cache performance and also to reduce the power consumption. Dybdahl et al. [5] initially proposed a write bypassing technique for LLC performance improvement. They identify the potential misses early and tend to bypass only those blocks. Zhang et al. [33] proposed a statistics based write bypassing technique (SBAC) which is based on the stats of the whole cache and instead of depending of only a single cache line. Kim et al. [12] proposed a inclusive bypass tag cache (IBTC) for NVM's. This method was presented to implement proper cache coherence for inclusive caches. Density proposed by Korgaonkar et al. [14] highlighted a reuse based cache bypassing technique that assigns a priority to any incoming block. This is helped through determining their liveness score which is important in determining weather a write-back to a LLC from the L2 cache is having the chance to be reused again in the near future. Wang et al. [29] proposed an adaptive LLC write bypassing technique for hybrid STT-RAM's which focuses on movement, bypassing and data placement based on the type of predetermined incoming write. Bagchi et. al [3] recently proposed, Performance Optimization and Endurance Management for Non-volatile Caches (POEM) which aimed at aggressively bypassing both write backs from the upper level cache and writes from the main memory while redistributing the remaining writes evenly among the cache lines. This not only helped them in gaining a significant improvement in performance but also helped with wear levelling and thereby enhancing the lifetime of the NVM based LLC.

7 Conclusion

Through our work we have highlighted the need for STT-RAM LLC's over SRAM based LLC's, but implementing purely STT-RAM LLC's come with their own set of issues like performance degradation and lifetime. Therefore, we propose PROLONG which is an innovative, power efficient and cost effective architecture that aims

Table 11: Normalized Lifetime of Write Intensive Workloads.

Workloads	PoLF	WAD	SWS	G-OAS(4)	SWWR	DWAWR	Pro(2%)	Pro(5%)	Pro(10%)	Pro(15%)
GAP										
bc	2.75	4.04	5.52	5.22	7.65	9.06	26.23	26.87	29.29	31.11
bfs	1.22	11.04	12.54	12.16	5.40	11.41	8.14	9.21	11.41	12.93
cc	1.39	4.89	6.41	6.10	6.10	9.28	26.74	28.07	30.82	32.73
pr	0.23	3.26	4.62	4.31	4.43	7.13	43.66	44.25	47.05	48.93
sssp	2.04	5.11	6.71	6.38	6.38	5.74	30.60	32.08	34.62	36.72
SPEC-2006										
cactus	6.45	6.45	7.84	7.84	3.77	13.79	10.00	12.24	14.58	16.20
gcc	2.81	3.39	5.17	4.57	3.39	6.40	10.24	12.27	15.09	16.56
Gems	2.61	1.41	2.85	2.61	3.59	13.35	5.61	6.39	8.52	9.90
lbm	1.52	5.95	7.44	7.01	4.30	14.10	6.59	7.23	9.43	11.02
leslie3d	0.87	2.21	3.58	3.35	6.68	8.43	16.04	16.92	19.33	20.89
libquantum	3.62	8.13	9.58	9.37	3.25	12.82	7.12	7.92	10.00	11.50
mcf	1.86	5.48	6.93	6.55	8.82	10.11	29.91	30.48	33.06	34.76
milc	4.81	3.43	4.81	4.53	6.52	8.49	34.71	35.33	38.03	39.83
soplex	1.96	1.56	2.97	2.77	0.78	5.69	22.07	22.93	25.30	27.14
sphinx	2.24	0.95	2.56	2.24	5.96	8.11	16.79	18.08	20.75	22.61
wrf	4.90	5.94	8.08	7.00	4.90	15.05	15.05	18.89	21.59	24.42
xalancbmk	1.25	3.05	4.52	4.11	4.11	13.76	30.43	31.28	33.88	35.68
zeusmp	2.27	4.65	7.14	5.88	5.88	3.45	7.14	11.11	13.92	15.38
Geomean All	2.00	3.80	5.57	5.16	4.63	9.14	15.88	17.62	20.43	22.32

at reducing the inter-set write variation and improve the lifetime of the STT-RAM LLC drastically without compromising on the performance of the system. Overall we have achieved a 89% reduction in inter-set write variation and 35 times lifetime improvement with a 0-2% improvement in performance for multicore systems. Therefore, PROLONG becomes a practical alternative to the traditional SRAM based LLC architectures as it can provide both the performance needed and also the required lifetime to implement an STT-RAM based LLC.

Acknowledgement

This work was supported in part by 2022 Qualcomm Faculty Award. The authors sincerely appreciate the valuable feedback and comments provided by our anonymous reviewers. We are also thankful to the members of CA-SIG Lab of IIT Ropar for all the helpful discussions and valuable feedback.

References

[1] Sukarn Agarwal and Hemangee K. Kapoor. 2017. Targeting inter set write variation to improve the lifetime of non-volatile cache using fellow sets. In *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. 1–6. https://doi.org/10.1109/VLSI-SoC.2017.8203453

[2] Sukarn Agarwal and Hemangee K. Kapoor. 2019. Improving the Lifetime of Non-Volatile Cache by Write Restriction. *IEEE Trans. Comput.* 68, 9 (2019), 1297–1312. https://doi.org/10.1109/TC.2019.2892424

[3] Aritra Bagchi, Dharamjeet, Ohm Rishabh, Manan Suri, and Preeti Ranjan Panda. 2024. POEM: Performance Optimization and Endurance Management for Non-volatile Caches. *ACM Trans. Des. Autom. Electron. Syst.* (mar 2024). https://doi.org/10.1145/3653452 Just Accepted.

[4] Scott Beamer, Krste Asanović, and David Patterson. 2017. The GAP Benchmark Suite. arXiv:1508.03619 [cs.DC]

[5] Haakon Dybdahl and Per Stenström. 2006. Enhancing Last-Level Cache Performance by Block Bypassing and Early Miss Determination. In *Advances in Computer Systems Architecture*, Chris Jesshope and Colin Egan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 52–66.

[6] Nathan Gober, Gino Chacon, Lei Wang, Paul V. Gratz, Daniel A. Jimenez, Elvira Teran, Seth Pugsley, and Jinchun Kim. 2022. The Championship Simulator: Architectural Simulation for Education and Competition. arXiv:2210.14324 [cs.AR]

[7] John L. Henning. 2006. SPEC CPU2006 benchmark descriptions. *SIGARCH Comput. Archit. News* 34, 4 (sep 2006), 1–17. https://doi.org/10.1145/1186736.1186737

[8] Adwait Jog, Asit K. Mishra, Cong Xu, Yuan Xie, Vijaykrishnan Narayanan, Ravishankar Iyer, and Chita R. Das. 2012. Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs. In *DAC Design Automation Conference 2012*. 243–252. https://doi.org/10.1145/2228360.2228406

[9] Mohammad Reza Jokar, Mohammad Arjomand, and Hamid Sarbazi-Azad. 2016. Sequoia: A High-Endurance NVM-Based Cache Architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24, 3 (2016), 954–967. https://doi.org/10.1109/TVLSI.2015.2420954

[10] Yongsoo Joo, Dimin Niu, Xiangyu Dong, Guangyu Sun, Naehyuck Chang, and Yuan Xie. 2010. Energy- and endurance-aware design of phase change memory caches. In *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. 136–141. https://doi.org/10.1109/DATE.2010.5457221

[11] Samira Manabi Khan, Yingying Tian, and Daniel A. Jimenez. 2010. Sampling Dead Block Prediction for Last-Level Caches. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '43)*. IEEE Computer Society, USA, 175–186. https://doi.org/10.1109/MICRO.2010.24

[12] Min Kyu Kim, Ju Hee Choi, Jong Wook Kwak, Seong Tae Jhang, and Chu Shik Jhon. 2015. Bypassing method for STT-RAM based inclusive last-level cache. In *Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems* (Prague, Czech Republic) *(RACS '15)*. Association for Computing Machinery, New York, NY, USA, 424–429. https://doi.org/10.1145/2811411.2811512

[13] Young-Bae Kim, Seung Ryul Lee, Dongsoo Lee, Chang Bum Lee, Man Chang, Ji Hyun Hur, Myoung-Jae Lee, Gyeong-Su Park, Chang Jung Kim, U-In Chung, In-Kyeong Yoo, and Kinam Kim. 2011. Bi-layered RRAM with unlimited endurance and extremely uniform switching. In *2011 Symposium on VLSI Technology - Digest of Technical Papers*. 52–53.

[14] Kunal Korgaonkar, Ishwar Bhati, Huichu Liu, Jayesh Gaur, Sasikanth Manipatruni, Sreenivas Subramoney, Tanay Karnik, Steven Swanson, Ian Young, and Hong Wang. 2018. Density Tradeoffs of Non-Volatile Memory as a Replacement for SRAM Based Last Level Cache. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 315–327. https://doi.org/10.1109/ISCA.2018.00035

[15] Kyle Kuan and Tosiron Adegbija. 2019. HALLS: An Energy-Efficient Highly Adaptable Last Level STT-RAM Cache for Multicore Systems. *IEEE Trans. Comput.* 68, 11 (nov 2019), 1623–1634. https://doi.org/10.1109/TC.2019.2918153

[16] Jianhua Li, Liang Shi, Chun Jason Xue, Chengmo Yang, and Yinlong Xu. 2011. Exploiting set-level write non-uniformity for energy-efficient NVM-based hybrid cache. In *2011 9th IEEE Symposium on Embedded Systems for Real-Time Multimedia*. 19–28. https://doi.org/10.1109/ESTIMedia.2011.6088521

[17] Ing-Chao Lin and Jeng-Nian Chiou. 2015. High-Endurance Hybrid Cache Design in CMP Architecture With Cache Partitioning and Access-Aware Policies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 10 (2015), 2149–2161. https://doi.org/10.1109/TVLSI.2014.2361150

[18] Sparsh Mittal. 2013. Using Cache-coloring to Mitigate Inter-set Write Variation in Non-volatile Caches. arXiv:1310.8494 [cs.AR]

[19] Sparsh Mittal and Jeffrey S. Vetter. 2014. EqualChance: Addressing Intra-set Write Variation to Increase Lifetime of Non-volatile Caches. In *2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads (INFLOW 14)*. USENIX Association, Broomfield, CO. https://www.usenix.org/conference/inflow14/workshop-program/presentation/mittal

[20] Sparsh Mittal and Jeffrey S. Vetter. 2015. AYUSH: Extending Lifetime of SRAM-NVM Way-Based Hybrid Caches Using Wear-Leveling. In *Proceedings of the 2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '15)*. IEEE Computer Society, USA, 112–121. https://doi.org/10.1109/MASCOTS.2015.29

[21] Sparsh Mittal, Jeffrey S. Vetter, and Dong Li. 2014. LastingNVCache: A Technique for Improving the Lifetime of Non-volatile Caches. In *2014 IEEE Computer Society Annual Symposium on VLSI*. 534–540. https://doi.org/10.1109/ISVLSI.2014.69

[22] Sparsh Mittal, Jeffrey S. Vetter, and Dong Li. 2015. A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-Volatile On-Chip Caches. *IEEE Transactions on Parallel and Distributed Systems* 26, 6 (2015), 1524–1537. https://doi.org/10.1109/TPDS.2014.2324563

[23] Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. 2007. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 40)*. 3–14.

[24] Moinuddin K. Qureshi, Sudhanva Gurumurthi, and Bipin Rajendran. 2011. *Phase change memory: From devices to systems*. 1–134. https://doi.org/10.2200/S00381ED1V01Y201109CAC018

[25] Puneet Saraf and Madhu Mutyam. 2019. Endurance enhancement of write-optimized STT-RAM caches. In *Proceedings of the International Symposium on Memory Systems* (Washington, District of Columbia, USA) *(MEMSYS '19)*. Association for Computing Machinery, New York, NY, USA, 101–113. https://doi.org/10.1145/3357526.3357538

[26] Jue Wang, Xiangyu Dong, and Yuan Xie. 2013. OAP: An obstruction-aware cache management policy for STT-RAM last-level caches. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 847–852. https://doi.org/10.7873/DATE.2013.179

[27] Jue Wang, Xiangyu Dong, Yuan Xie, and Norman P. Jouppi. 2013. i2WAP: Improving non-volatile cache lifetime by reducing inter- and intra-set write variations. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 234–245. https://doi.org/10.1109/HPCA.2013.6522322

[28] Shuai Wang, Guangshan Duan, Yupeng Li, and Qianhao Dong. 2017. Word- and Partition-Level Write Variation Reduction for Improving Non-Volatile Cache Lifetime. *ACM Trans. Des. Autom. Electron. Syst.* 23, 1, Article 4 (aug 2017), 18 pages. https://doi.org/10.1145/3084690

[29] Zhe Wang, Daniel A. Jiménez, Cong Xu, Guangyu Sun, and Yuan Xie. 2014. Adaptive placement and migration policy for an STT-RAM-based hybrid cache. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. 13–24. https://doi.org/10.1109/HPCA.2014.6835933

[30] H.-S. P. Wong, C. Ahn, M. Beauchamp, J. Cao, H.-Y. Chen, W. C. Chen, S. B. Eryilmaz, S. W. Fong, W. Hwang, J. A. Incorvia, R. Islam, Z. Jiang, H. Li, S. Liu, C. Neumann, K. Okabe, S. Qin, Y. C. Shen, M. Shi, J. Sohn, M. Tung, W. Wan, X. Wu, Y. Wu, S. Yu, Z. Yu, and X. Zheng. Accessed June 6, 2022. Stanford Memory Trends. https://nano.stanford.edu/stanford-memory-trends.

[31] Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, and Yuan Xie. 2009. Hybrid cache architecture with disparate memory technologies. In *Proceedings of the 36th Annual International Symposium on Computer Architecture* (Austin, TX, USA) *(ISCA '09)*. Association for Computing Machinery, New York, NY, USA, 34–45. https://doi.org/10.1145/1555754.1555761

[32] Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, and Yuan Xie. 2009. Hybrid cache architecture with disparate memory technologies. *SIGARCH Comput. Archit. News* 37, 3 (jun 2009), 34–45. https://doi.org/10.1145/1555815.1555761

[33] Chao Zhang, Guangyu Sun, Peng Li, Tao Wang, Dimin Niu, and Yiran Chen. 2014. SBAC: A statistics based cache bypassing method for asymmetric-access caches. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 345–350. https://doi.org/10.1145/2627369.2627611

[34] Lijie Zhang, Yen-Ya Hsu, Frederick T Chen, Heng-Yuan Lee, Yu-Sheng Chen, Wei-Su Chen, Pei-Yi Gu, Wen-Hsing Liu, Shun-Min Wang, Chen-Han Tsai, Ru Huang, and Ming-Jinn Tsai. 2011. Experimental investigation of the reliability issue of RRAM based on high resistance state conduction. *Nanotechnology* 22, 25 (may 2011), 254016. https://doi.org/10.1088/0957-4484/22/25/254016