Enhancing Endurance of NVMs by Coarse-To-Fine Grained Write Reduction and Intra-line Wear Leveling

Thesis submitted in partial fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

Submitted by

Arijit Nath

Under the guidance of

Prof. Hemangee K. Kapoor

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

January, 2023

Dedicated to

Lord Shiva

and

My Dearest Parents

Declaration

I certify that:

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisor.
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.
- I am fully aware that my thesis supervisor is not in a position to check for any possible instance of plagiarism within this submitted work.

Date: January, 2023
Place: Guwahati

(Arijit Nath)

Acknowledgements

The pursuit of Ph.D. has provided me with immense opportunities for learning and development on both professional and personal levels. It is a humble effort to express my gratitude to the individuals who have supported and contributed in various ways throughout my Ph.D. journey. First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Hemangee K. Kapoor, for her excellent guidance, inspiration, and significant help over the past few years. She has been gracious enough to supply all that is required, including essential resources and an environment suitable for carrying out research without any difficulties. Thank you, Madam Hemangee, for correcting my mistakes and constantly motivating me to focus on the Ph.D. works. I am also thankful to her for giving me an opportunity to work collaboratively with other students, including B.Tech, M. Tech, and junior Ph.D. students. It has helped to enhance my knowledge and research skills.

I am grateful to my doctoral committee members: Prof. Diganta Goswami, Dr. Aryabartta Sahu, and Dr. Sonali Chauhan, for their constructive suggestions for preparing my thesis. Additionally, I would like to thank Prof. Jatindra Kumar Deka, the Head of the Department of Computer Science and Engineering, and other faculty members for their constant assistance and support. Furthermore, I would also like to thank Prof. Kalpesh Kapoor for his numerous academic/non-academic suggestions.

Working with Dr. Shirshendu Das, Dr. Shounak Chakraborty, Dr. Sukarn Agarwal, Dr. Khushboo Rani, Dr. Palash Das, Dr. Sheel Sindhu Manohar, Aswathy, Imlijungla, Neeraj, Swati, Rishabh, Lt. Col. Alankar V. Umdekar, Neelkamal, Manik and Nishant was an honor during my doctoral studies. Innumerable times, our knowledge-sharing and technical talks aided in my research.

During Ph.D., I was fortunate to have many wonderful friends who added colors to my Ph.D. life. I feel extremely lucky to meet people like Pradeep Bhale, Ujjwal Da, Pawan, Akshay, Chinmaya Sir, Harish Da, Akansha, Divya, Dipojjwal, Tamen, Ritupan, Nilotpal, and Maithli. Also, special thanks to my juniors: Prachurya, Swati, Neeraj, Neel Madhab, Somyadeep, Rishabh for being with me during both happy and tough times. I will treasure the memories I have with these people for the rest of my life.

I would like to express my sincere gratitude to Mr. Raktajit Pathak, Mr. Nanu Alan Kachari, Mr. Bhriguraj Borah, Mr. Monojit Bhattacharjee, Mrs. Gauri Deori, Mr. Gaurish Majumdar, and all other department staff for their assistance throughout my time at IIT Guwahati in various capacities and at multiple times. Furthermore, I would like to thank the student affairs section for providing an on-campus accommodation facility to the students. Additionally, I want to thank the security officers, janitors, hostel mess staff, and canteen personnel for making my time on the IIT Guwahati campus comfortable.

Finally, I would like to express my profound gratitude to my family members. My parents, the two most important persons of my life, have been extremely supportive throughout the journey. Their unconditional love and immense motivation gave me the needed strength to deal with all the hurdles with confidence. My sister (Sanchayita) and brother (Arindam) made my journey smooth not only by inspiring me throughout the Ph.D. journey but also by taking great care of my aging parents. Finally, I am honored to pursue my Ph.D. from an Institute located in my home state Assam. The memories of its beautiful environment and the lovely local people will remain in my heart forever.

Date: January 30, 2023
Place: Guwahati

Arijit Nath

Certificate

This is to certify that this thesis entitled,**"Enhancing Endurance of NVMs by Coarse-To-Fine Grained Write Reduction and Intra-line Wear Leveling"**, being submitted by **Arijit Nath**, to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for partial fulfillment of the award of the degree of Doctor of Philosophy, is a bonafide work carried out by him under my supervision and guidance. The thesis, in my opinion, is worthy of consideration for award of the degree of Doctor of Philosophy in accordance with the regulation of the institute. To the best of my knowledge, it has not been submitted elsewhere for the award of the degree.

Date:
Place: Guwahati

.....

Dr. Hemangee K. Kapoor
Professor
Department of Computer Science and Engineering
IIT Guwahati

Abstract

The unprecedented development in the processing speed of the Chip Multi-Processor (CMP) and the rise of modern data-intensive applications impose high pressure on the memory subsystem. Due to the large working set data and low temporal locality, these applications can not utilize cache memories fully, leading to more cache misses. It significantly increases the main memory footprint and necessitates designing of energy-efficient and high capacity main memory. Unfortunately, the traditional memory systems, built predominantly using DRAM are not scalable to the low nanometer regime. At this need of the hour, the Emerging Non-Volatile Memories (NVMs) like PCM, STT-RAM, ReRAM offer fascinating features like high density and low leakage power that are useful for building high capacity and energy-efficient memory systems. However, NVMs have asymmetric read/write operations, where writes are costly in terms of latency and energy. Also, frequent write operations to the NVM cells tend to wear out the memory cells, leading to a shortened memory lifetime. Furthermore, the non-volatility feature of the NVMs leads to security threats related to data confidentiality that were never encountered before. NVMs retain data even after the system is powered down. Hence, an attacker having physical access to the NVM DIMM can easily stream out the sensitive data stored in the NVM. Researchers have proposed encryption-based techniques to protect the sensitive NVM content. However, encryption algorithms put high randomization in the encrypted data, which leads to enormous bit-flips when the encrypted data is written in the NVM arrays. Hence, the lifetime issue of the NVM devices is further complicated by encryption-induced bit-flip spikes.

The contributions to this thesis revolve around designing policies to reduce write operations in the NVMs. In addition to extending the lifespan of standard NVMs, the contributions also cover reducing bit-flips caused by encryption and providing a strong security guarantee against *data confidentiality*-based attacks on NVMs. In particular, we have proposed policy to reduce write-back traffic of the evicted blocks from the Last Level Cache (LLC) to the NVM. While it reduces writes at a *cache block* level, the other contributions of the thesis focus on building efficient compression and encoding techniques to reduce bit-flips, which are *fine granularity* write operations and proposing wear leveling algorithms to even out the bit-flips pressure across the memory space. In the last two contributions, we have designed a partial encryption-based encoding policy and propose techniques utilizing various compression and encoding techniques to reduce encryption-induced bit-flips while ensuring confidentiality in NVMs. The proposed techniques show significant improvement in lifetime, energy-efficiency and performance compared to the state-of-the-art techniques.

Contents

List of Figures

List of Tables

1 Introduction

Moore's law [1], which is regarded as one of the revolutionary theories of the 21st century, has been the driving force behind the continuous development of semiconductor technology over the past few decades. According to this law, the number of transistors in a chip doubles about every two years. Higher transistor counts helped in improving performance of the single-core machines prevalent in the early years of development of computing systems by increasing their clock frequency. But, with the end of Dennard scaling (Power wall) [2] and limitations of Instruction Level Parallelism (ILP) [3], the performance of the single-core systems came to a standstill, primarily due to the excessive power consumption at higher frequencies. It led to the rise of multi-core systems commonly known as the Chip Multi Processors (CMP) [4]. CMPs offer high performance with low power consumption by keeping multiple cores within the same processing die. As the transistor's size continues to shrink further, the number of on-chip cores is also increasing proportionally, resulting in an epoch-making revolution in the processing speed of the CMPs. For example, Intel Xeon Phi [5] AMD EPYC [6], and Ampere Altra [7] are some contemporary CMPs that feature up to 120 cores in the processing chip.

The unprecedented growth in computational power facilitates the concurrent execution of many applications. Due to the frequent interaction of these applications with the memory, the contention in the shared main memory has increased [8], [9], [10]. Unfortunately, the memory system's speed increases at a fairly slow rate compared to processing speed, mainly due to the narrow width of the processor-memory off-chip bus ((e.g., standard Double Data Rate, (DDR), memories use a 64-bit memory channel [11], [12]).). At the same time, the limited capacity of the current memory systems leads to frequent page faults,

causing further slowdown of the system performance. It creates a severe performance bottleneck while executing such applications. To reduce the performance degradation due to the processor-memory speed gap, fast on-chip cache memories based on temporal/spatial data locality [13], [14] appeared as an effective innovation. However, with the advent of modern data-intensive applications, it has become increasingly difficult for the caches to mitigate the effects of main memory that is sluggish and has limited capacity. The voluminous data access patterns of these applications [15], [16], [17] diminish the effectiveness of caches due to their limited capacity. As a result, execution of these applications increases the main memory footprint. Not only that, frequent memory accesses also account for high energy consumption [18], which has become a matter of concern in large workstations and data centers. Consequently, designing fast, high-capacity, and energy-efficient memory systems is becoming the need of the hour in the current modern digital era.

At this critical juncture, the conventional main memory, built predominantly using DRAM technology, is facing severe challenges in terms of scalability and energy efficiency [19],[20],[21]. DRAM scaling below 30nm technology node is difficult, as factors like high refresh energy, write recovery time (tWR), and Variable Retention Time become more prominent [19]. In light of the difficulties experienced by DRAM, researchers are investigating other memory technologies as alternatives to DRAM. Currently, the emerging Non-Volatile Memories (NVM) [21], [22], [23], [24], [25], [26] appear as a strong contender of DRAM in the main memory. NVMs show exciting features like high density, near-zero leakage power, and non-volatility.

1.1 Modern CMPs and Memory System Trends

CMPs attach a large number of cores on the same die. In terms of power budget and processing capability, these cores could be homogeneous or heterogeneous (or a combination of both) [4]. While homogeneous cores can execute computationally intensive applications by dividing the load of heavy computations among the cores, heterogeneous cores enable the concurrent execution of many applications with a wide variety and purposes (multimedia applications, compute-intensive applications like Convolutional Neural Networks (CNN), and less computation involved operations like spreadsheets and word processing). As all these cores share the main memory, the interference in the main memory increases dramatically. It leads to increased demand for memory capacity and bandwidth along with the quality of service to provide fairness among the competing cores [27], [28], [29], [30], [31].

The nature of today's data-intensive applications also contributes significantly to the stress placed on the memory sub-system. Emerging applications like IoT applications, Image/Video classification, speech recognition, recommendation systems, etc., process large datasets [32], [33], [34]. Most of these applications use machine learning and data analytics to extract resourceful information from large data sets. For example, many Internet of

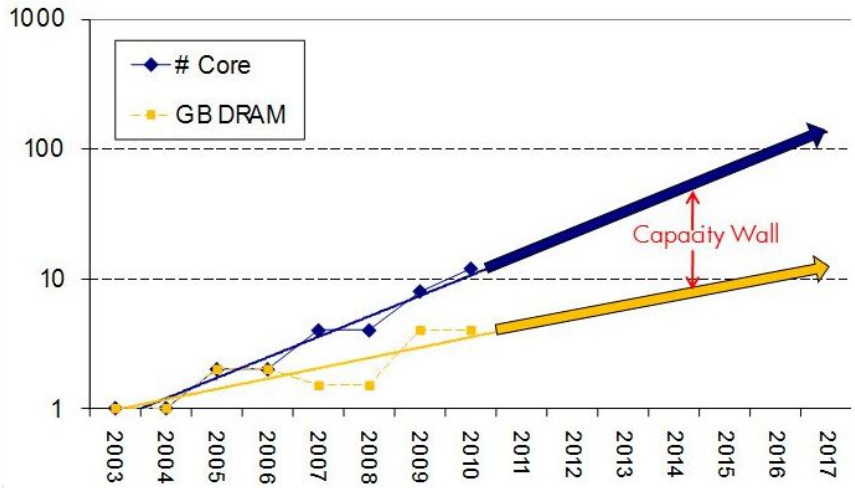


Figure 1.1: *Memory Capacity Wall [38]*

Things (IoT) applications take in voluminous data from various sensors and process that data using some learning models to arrive at key conclusions (e.g., monitoring and response of patient's health, navigation of self-driving cars). As a result, these applications exhibit a large working set size, which is difficult to fit into the limited memory. It leads to frequent page faults and degrades the system performance. Additionally, several of these applications show a low locality of reference [35], [36], [37]. Hence, conventional cache memories optimized to improve performance end up being less effective for such applications. It leads to cache misses and a corresponding increase in the main memory footprint.

To meet the increasing demand for memory capacity, the traditional DRAM-based main memory is undergoing a tough scaling process. Due to the limitations of DRAM scaling, the rate of increase in DRAM density has reached saturation. The growing discrepancy between the number of cores and memory capacity per socket (depicted in Figure 1.1) has led to the rise of one more significantly important wall known as *Memory Capacity Wall* [38]. As predicted in [38], the memory capacity per core is supposed to drop by 30%, especially for commodity servers in about every two years. Hence, developing high capacity memory system has paramount importance at this pivotal time.

1.2 Technologies Used for Main Memory and Challenges

Over the years, DRAM has served as the de facto standard for creating main memory. However, the possibility of deploying several newly emerging non-volatile memories has also been investigated in recent years. Below, we provide a quick overview of various memory

technologies and the challenges associated with their implementation.

1.2.1 DRAM and its Challenges

Fundamentally, a DRAM cell is composed of a *storage capacitor* and *access transistor*. Each bit of data in a DRAM cell is represented by the amount of charge on its capacitor. The data stored in a DRAM cell is represented as logic 1(0) if the charge in the capacitor is above(below) a certain threshold. Due to the leaky nature of the capacitor, DRAM cells tend to lose charge over time, which could eventually cause the bits stored in the cells to be flipped. Hence, DRAM cells need periodic refreshes to guarantee that their cells contain correct binary values.

DRAM account for high refresh and leakage energy consumption. Furthermore, scaling DRAM to a low nanometer regime (below 10 nm) is challenging as DRAM cells become leakier and start containing erroneous bit values.

1.2.2 Emerging Non-Volatile Memories (NVM) and Challenges

With the DRAM scaling nearly hitting saturation, the emergence of some Non-Volatile Memory (NVM) technologies has drawn the attention of researchers. NVMs include Spin Transfer Torque Random Access Memory (STT-RAM), Resistive Random Access Memory (ReRAM), Phase Change Random Access Memory (PCRAM), and many others. In contrast to DRAM, which represents data as the charge held in its capacitor, NVMs store the bit information in their cells in the form of resistance. They offer variable resistance to store the bit information by changing the material properties of the cells. For instance, a PCM cell exhibits different resistance if its constituent chalcogenide material is heated and cooled at different rates. Logic 1(0) is represented by the cells' high(low) resistance state.

NVMs provide many exciting features like non-volatility, high density, and low leakage power consumption. Non-volatility features aid in minimizing power and system failures, improving checkpointing, and speeding up application startup. Additionally, properties like high density and low leakage power consumption support constructing large capacity and energy-efficient main memory systems.

Despite offering many benefits, adopting NVMs as a mainstream main memory standard is still under question. Along with the advantages, NVMs posses certain downsides that can not be underestimated. NVMs have asymmetric costs in read and write operations, whereas writes are costlier in terms of latency and energy consumption. The costly writes also cause significant wear to the NVM cells, leading to their early wear out. As a result, NVMs have limited write endurance and a shortened lifetime. Furthermore, the non-volatility feature opens the door to major security risks, as the data stored in the NVM cells is retained even after the system is powered down. An attacker having physical access to an NVM DIMM can easily stream out the sensitive data stored in the NVM. This type of attack is known

as the *stolen DIMM attack* [39]. Researchers have proposed encryption-based techniques to protect data against such attacks. However, there are serious unintended consequences of using encryption to provide security. The majority of common encryption methods exhibit the *diffusion* property [39, 40], which imposes significant randomization in the generated cipher text (encrypted text). It leads to a spike in bit-flips when the encrypted data blocks are written in NVM arrays. As a result, the longevity issue with NVM devices is further complicated by the dramatically increased write activity.

1.3 Motivations

NVMs are widely regarded as a competitive alternative to DRAM in the main memory due to their high density, low leakage power consumption, and non-volatility features. However, their direct adoption is hampered by a few intrinsic drawbacks, such as low cell endurance, high write latency, and high write energy. Furthermore, due to the prolonged data retention property, NVMs are vulnerable to confidentiality-based attacks.

Researchers have explored various architectural options to construct NVM-based memory systems to utilize their potential in the best possible way. Two such design choices are *Hybrid Main Memory* and *NVM-based Main Memory* systems. Hybrid memories are built using a small portion of DRAM accompanied by a larger NVM portion to utilize the benefits of both memory media. In such memory systems, DRAM provides latency benefits by storing the performance-critical blocks, and NVM offers density benefits by keeping the other blocks. On the other hand, architects have also proposed solutions at the device and architecture levels to integrate NVM in main memory fully. Although both these design choices open up ample opportunities to build efficient main memory systems for next-generation computers, the primary issue in deploying NVM revolves around the very nature of their write operations. NVMs show asymmetric read/write operations where writes are much costlier than the reads in terms of latency and energy. Frequent writes can degrade the system performance and cause damage to the memory cells. As a result, NVMs exhibit poor write endurance compared to traditional DRAM-based memories. For example, the write endurance of PCM [24] and ReRAM [26] are 10^8 and 10^{11} writes, respectively, whereas the endurance of DRAM-based memories is quite high ($> 10^{16}$). Also, writes contribute to high energy consumption, as they play a dominant role in the dynamic energy consumption in the NVMs. Hence, reducing write activities is a great way to improve lifetime and minimize energy consumption in NVMs.

In this thesis, we have proposed techniques to reduce writes at two granularities, 1) *Coarse granularity* (at block level) and 2)*Fine granularity* (at bit level). We propose our contributions for PCM-based non-volatile memory due to its high density and excellent scalability in CMOS fabrication [21, 41]. Since the proposed techniques are applied on the blocks incoming to NVM before actually writing them in memory, they can reduce and

uniformly distribute writes in NVMs, irrespective of the memory technology. Hence, they are equally adaptable to the other NVMs as well.

The rest of the thesis considers PCM as the NVM.

Evictions of the cache blocks from the Last Level Cache (LLC) contribute to a large fraction of writes in the PCM-based main memory. The incessant flow of write-backs coming to PCM can damage the memory cells and lead to poor system performance. It encourages us to develop efficient techniques that reduce costly write-backs to PCM devices.

Write activities in PCM could also be reduced at the cell level by using avenues like compression and encoding. If the evicted blocks coming to PCM are compressed before writing, a lot of bit-flips can be saved when the compressed data is written over the PCM arrays. On the other hand, encoding techniques change the data bits written in PCM to formats that reduce bit-flips. Hence, encoding performed over the compressed blocks can further reduce bit-flips. Apart from reducing bit-flips, the distribution of writes within the memory lines also plays an important role in the lifetime of PCM. As the compressed data blocks are written on one side of the memory lines, the cells of the corresponding side face more bit-flips than the other side, creating write hot spots within the memory lines. It leads to early wear out of the cells facing more bit-flips. Hence, the lifetime of PCMs can be improved further if the uneven distribution of bit-flips could be balanced by the periodic shifting of the writing position of the compressed blocks within the memory lines. In this direction, we have proposed novel compression and encoding schemes to reduce bit-flips along with an intra-line wear leveling technique to disperse the bit-flips uniformly over the PCM cells.

Furthermore, non-volatility feature makes the NVMs vulnerable to *confidentiality*-based attacks. Due to non-volatility, the data stored in NVMs remains persistent for a long duration, even after the system is powered off. Hence, a malicious attacker having physical access to NVM DIMM can easily stream out confidential data. This type of attack, commonly known as *Stolen DIMM* attacks can be prevented if the data in NVM is protected using encryption. But, security provisioning through encryption does not go hand in hand with the endurance issue of the NVMs. In order to obfuscate the attackers from retrieving the data content, encryption algorithms put high randomization in the generated cipher text. When the encrypted cipher blocks are written, the NVM cells encounter a lot of bit-flips. This phenomenon is known as the *Avalanche Effect*. The increase in bit-flips worsens the endurance of NVMs. Bit-flips reduction methods like [42–44] optimized for un-encrypted memories become ineffective in this context due to the high spike in bit-flips. Therefore, bit-flip reduction in the presence of encryption is more challenging and needs careful handling in order to maintain durability of NVMs. Our final two contributions are related to reducing bit-flips in encrypted memory with an aim to improve their endurance.

In other words, the proposed methods aim to extend the useful life of NVMs while also guaranteeing the safety of data stored on them.

The primary goal of the thesis is to decrease expensive writes (at the block and bit levels) in order to extend the lifespan of PCM. In this regard, we propose our techniques for both DRAM-PCM based *Hybrid memory* and *PCM*-based main memory. With strict adherence to this objective, our contributions also include ensuring data security through encryption while lowering the encryption-induced surge in bit-flips. We have made different contributions in the following directions.

- We develop a reuse distance-aware selective victim cache-based policy to reduce write-backs to the PCM component of a DRAM-PCM hybrid main memory. Victim cache holds the performance-critical blocks evicted from LLC and prevents them from getting written back. To reduce the PCM write-backs further, we propose two techniques based on *Block Placement* and *Partitioning* of the victim cache.

- We suggest a comprehensive approach that combines compression and encoding to lessen bit-flips and an intra-line wear leveling strategy to distribute the bit-flips throughout the PCM cells evenly.

- We propose a partial encryption-based encoding technique to reduce bit-flips in encrypted PCM-based main memory.

- We provide new strategies that utilize the benefits of various compression and data encoding techniques to reduce bit-flips in encrypted PCM.

1.4 Thesis Objectives

The main objective of the thesis is to improve the endurance of PCM-based main memories. The proposed techniques are designed to reduce as well as evenly distribute costly writes in PCM memory space. Reducing writes also aids in energy consumption minimization and improved performance. The more specific goals of the thesis are outlined below.

1. **Improving Endurance of PCM using Write Reduction :** We aim to design policies that reduce writes at the PCM cells. Our techniques are designed to reduce writes both at *block* level and *bit* level. Our proposed *block* level techniques reduce the write-back traffic from LLC to PCM-based main memory. In contrast, the proposed *bit*-level techniques reduce bit-flips in the PCM cells using compression and encoding. It helps in improving the PCM lifetime as the cells tend to wear out much slowly if write activities are reduced considerably.

2. **Improving Endurance of PCM using Write Distribution :** During workload execution, some memory cells are exposed to writing more often than the other cells, leading to non-uniform distribution of bit-flips within the memory lines. As a result, the cells experiencing more writes tend to wear out much more quickly than the other cells, which degrades memory lifetime. Towards this, we have proposed an intra-line wear leveling technique that balances the uneven write pressure within the memory lines by periodically shifting the data content.

3. **Reducing Energy Consumption :** In PCM-based main memory system, writes constitute a major portion in dynamic energy consumption. Reducing writes using our proposed techniques can also serve the purpose of lowering energy consumption in PCM.

4. **Improving System Performance :** Our techniques also remain successful in improving system performance by reducing writes. Write reduction can boost system performance as the contention for the subsequent read operations are reduced significantly.

5. **Providing Confidentiality Guarantee to the PCM data :** Due to prolonged data retention, PCM data can be stolen by streaming out confidential data from the PCM DIMM. We propose techniques based on partial encryption and a combination of compression and encoding that protect PCM data. These techniques are also optimized to reduce the encryption-induced bit-flips to provide longevity to the PCM in the presence of encryption.

6. **Exploring Types of Memory Systems**: PCM can be utilized in one of two ways, either as (a) hybrid memory alongside DRAM or as (b) complete NVM-based main memory. We propose techniques for both types of memory system to improve their efficiency and usability.

1.5 Thesis Contributions

PCM-based main memories suffer from weak write endurance, leading to their early wear out on facing frequent writes. Furthermore, non-volatility feature of such memories exposes them to data confidentiality threats that needs further consideration. Encryption based approaches, though provide solid security guarantee, often complicates the endurance issue of PCM further as they increase bit-flips by imposing high degree of randomness in the encrypted data. This thesis aims at improving the endurance hence lifetime of PCM-based main memories using write reduction and write distribution techniques. We propose coarse-grained (victim caching) and fine-grained (compression and partial encryption scheme) write

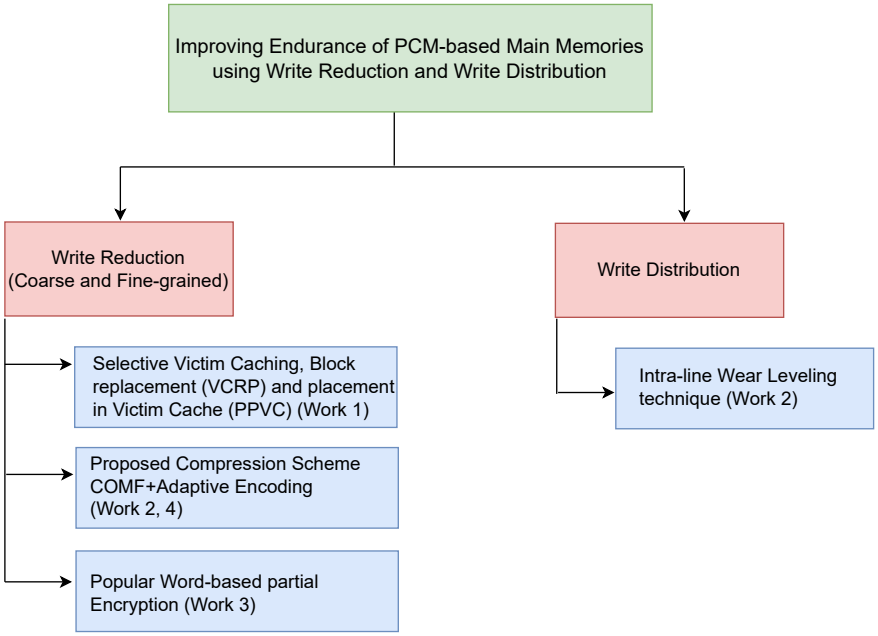


Figure 1.2: *High level digram illustrating thesis contributions*

reduction techniques and intra-line wear leveling technique to uniformly distribute the writes through out the memory space. The effect of fine-grained compression-based approaches have been demonstrated for encrypted and non-encrypted memories. Coarse-grained victim cache based approach is proposed assuming the underlying hybrid memories as unencrypted. However, it is supposed to show good results for encrypted PCM as well, as reduced write-backs tend to relieve write pressure in encrypted PCM; thereby improving endurance. In figure 1.2, we give a top level picture of the thesis before diving into the brief description of the contributions.

In this section, we present a brief overview of the contributions of the thesis. The first two contributions offer ways to reduce writes at the block and bit granularity, respectively. In contrast, the final two contributions focus on minimizing bit-flips while protecting PCM data via encryption.

1.5.1 Write Reduction Using Selective Victim Caching (Contribution 1)

Hybrid memories utilize the benefits of the low latency of DRAM to improve performance and the high density of PCM to increase main memory capacity. However, the lazy and energy-intensive write-backs of the evicted blocks from LLC to the PCM component of hybrid memory limit system performance as well as its longevity. Therefore, the effectiveness

of hybrid memories is largely impacted by the replacement policies of the LLC. The traditional cache replacement policies like *Least Recently Use* (LRU) are optimized to increase hit rate in the LLC. But, these policies are oblivious of PCM's read/write disparity, where PCM writes are much costlier than the reads in terms of latency and energy. Hence, they do not bring similar improvements in performance in the case of hybrid memories. Additionally, the costly PCM write-backs tend to damage the PCM component of the hybrid memory, leading to an eventual degradation of memory lifetime.

In this contribution, we propose two block-level write reduction techniques for DRAM-PCM-based hybrid main memories. The proposed techniques are based on using a Victim Cache (VC) [45] that sits between LLC and main memory. The VC keeps the performance-critical blocks evicted from the LLC. The criticality of a block is determined by its *reuse distance* [1]. We use the idea of reuse distance to identify the blocks with a history of short reuse distance usage during their LLC residency. Since short reuse distance of a block indicates its high temporal locality, the blocks that have a high *Frequency of short reuse distance* during their LLC residency can be classified as the *critical blocks*. These blocks have a high chance of being re-used again in the forthcoming memory references. We therefore keep such *critical blocks* in the VC rather than writing them back to hybrid memory. Based on this idea of selective victim caching, we propose two techniques that work on the *Replacement Policy* and *Partitioning* of the VC to reduce the PCM write-backs further. We briefly discuss these techniques below.

- **Victim Cache Replacement Policy (VCRP)** : VCRP prioritizes the eviction of the DRAM blocks[2] over the PCM blocks. Prior to eviction, VCRP searches two LRU positions (LRU and LRU-1 in the LRU stack) in the VC. If a DRAM block is found during the search, it is preferentially evicted over a PCM block. Since evicting PCM blocks is more expensive than evicting DRAM blocks, PCM blocks are preferred to be kept in the victim cache.

- **Prioritized Partitioning of Victim Cache (PPVC)**: PPVC logically partitions the VC, giving more share to the critical PCM blocks and comparatively lesser share to the critical DRAM blocks. Thus, PCM blocks will get more space in the VC than the DRAM blocks. While it might seem prudent to allocate all available VC space to PCM blocks to reduce PCM write-backs, reserving some space for DRAM blocks is necessary to prevent the abrupt degradation of system performance due to the eviction of some important critical DRAM blocks.

[1] Reuse distance of a cache block is the number of distinct accesses between two consecutive accesses to the block during its residency in the cache.

[2] A cache block is termed as DRAM(PCM) block if it corresponds to DRAM(PCM) partition in the hybrid main memory

We compare our techniques (VCRP and PPVC) with two baselines (Baseline 1: Without VC, Baseline 2: LLC+VC without reuse distance), and two state-of-the-art techniques: VAIL [46] and WBAR [47]. On average, <VCRP, PPVC> reduce PCM write-backs by <11.27%, 10.82%>, improve performance by <5.8%, 5.65%> and reduces energy consumption by <11.73%, 11.52%> over Baseline 1 (without VC). The details of this work are discussed in Chapter 3.

1.5.2 SWEL-COFAE : Write Reduction Using Compression and Adaptive Encoding Augmented by Wear Leveling (Contribution 2)

When the blocks evicted from the LLC are written in PCM, it causes bit-flips in the PCM cells. These bit-flips are *fine granularity* bit-level write operations that degrade the memory lifetime and lead to high energy consumption. Hence, reducing write activities at the bit level (i.e., bit-flips) is necessary to improve durability and minimize energy consumption in PCM. In this contribution, we have proposed approaches based on compression and encoding to cut down on the bit-flips in the PCM-based main memory.

Compression reduces the effective size of the blocks written in PCM. When the reduced blocks are written in the PCM arrays, many cells remain unexposed to writing. Hence, bit-flips are reduced in the unexposed cells. On the other hand, encoding techniques convert the compressed blocks to formats that reduce bit-flips in the PCM cells. We have devised a novel compression algorithm based on the word-level similarity among the words within the cache blocks. On top of it, we have applied an adaptive encoding technique to reduce bit-flips further. Although reducing bit-flips improves PCM lifetime, writing of the compressed blocks on the same side of the memory lines[1] makes that side heavier in terms of bit-flips compared to the unexposed side. In other words, the cells corresponding to the exposed part of the memory line experience more bit-flips and tend to wear out faster than the other cells of the memory line. The lifetime of PCM can be improved further if the uneven bit-flips can be balanced. We have proposed two intra-line wear leveling algorithms in an incremental fashion to ease out the imbalance created by uneven bit-flips. We term our integrated approach of compression, encoding, and wear leveling as SWEL-COFAE. We briefly explain the working of SWEL-COFAE below.

- **Proposed Compression Technique (COMF)**: Our compression technique, called COMF, exploits the data redundancy that exists among the words within the cache blocks. The cache blocks evicted from LLC and approaching PCM contain many words that repeat with high frequency. Our experiments using several PARSEC and SPEC 2006 benchmarks reveal that, on average, the percentage of words repeating

[1] A memory line refers to an array of PCM cells where a cache block gets fitted.

between 11-16, 6-10, 3-5, and 0-2 times are 53%, 27%, 14%, and 6%, respectively, in a cache block containing 16 words (with size 64 bytes and word width 4 bytes). It indicates that the words within a cache block are frequently repeated.

COMF compresses the blocks by removing the repeated occurrences (except its first occurrence in the cache block) of the Most Frequently Occurring Word (MFW) within the cache lines. It maintains certain meta-data (explained in detail in Chapter 4) that help during decompression. Since compression/decompression incurs latency, we compress a block if the MFW frequency exceeds a predefined threshold (th). The analysis of Compression/ Decompression in terms of area, latency, and power is discussed in Chapter 4.

- **Adaptive Encoding technique, COFAE :** In order to reduce bit-flips further, we apply a FNW [43]-based adaptive encoding strategy on the compressed blocks produced by COMF. The combination of COMF and adaptive encoding is termed COFAE. FNW divides the data blocks into equal-sized partitions and assigns a tag bit to each of them. If the bit-flips in a partition is more than half of the partition size (termed as *Encoding Granularity (*$Gran_{fixed}$*)*), then the bits are written in inverted form, and the corresponding tag bit is set to 1. Else, the bits are written as it is and the corresponding tag bit is set to 0. FNW bounds the maximum number of bit-flips in a partition to half of *Encoding Granularity*. FNW reduces bit-flips to a greater extent when the *Encoding Granularity* becomes finer (smaller), as supported by [43, 44, 48]. While fine *Encoding Granularity* is desirable, it involves more tag bits to achieve fine granularity, resulting in large storage overhead.

  In the proposed adaptive encoding approach, we encode the compressed blocks at finer *Encoding Granularity* by assigning the tag bits to the compressed data bits during runtime. Hence, the number of data bits represented by a tag bit becomes less, resulting in finer *Encoding Granularity*. Note that we do not have to increase the number of tag bits to achieve finer granularity. Since the encoding granularity gets adjusted dynamically in real time depending on the size of the compressed block, we term it *Adaptive Encoding Granularity (*$Gran_{adapt}$*)*. Our encoding approach leads to a further reduction in the bit-flips.

- **Proposed Intra-line Wear Leveling Techniques :** We propose two intra-line wear leveling techniques to balance the uneven bit-flip distribution in the PCM memory lines.

  1. Orientation-based Wear Leveling Technique : In this technique, we periodically change the direction (from left-to-right and right-to-left and vice-versa) of writing the compressed blocks in the memory lines after a specific number of writes. An orientation bit indicates the orientation of writing the compressed blocks. In particular,

orientation bit value 1(0) indicates left-to-right(right-to-left) orientation of writing.

2. Stride-based wear leveling technique : Orientation-based wear leveling introduces more bit-flips towards the extreme ends of the memory lines. However, the PCM cells corresponding to the middle portion of the memory lines remain unexposed most of the time. It leads to an uneven distribution of bit-flips inside the memory lines where the cells towards the ends face more bit-flips than the middle cells.

We propose a wear leveling technique, called *stride-based* wear leveling, that balances the bit-flip pressure within the memory lines by periodically shifting the writing position of the compressed lines, decided by a stride distance. It leads to uniform distribution of bit-flips within the memory lines and further enhances the PCM lifetime.

We have compared the effectiveness of our proposed policy SWEL-COFAE with baseline (DCW [42]) and four existing techniques [49], [50], [51] and READ [44]. Experimental results show that SWEL-COFAE reduces bit-flips by 59%; reduces energy consumption by 61% and improves lifetime by 101% over the baseline DCW technique. We have given full description of this work in Chapter 4.

1.5.3 Pop-Crypt : Reducing Encryption Overhead in NVMs by Identification of Popular Words (Contribution 3)

In this contribution, we propose a partial encryption scheme called Pop-Crypt that provides a robust security guarantee to the PCM data against *stolen DIMM* attacks and minimizes the encryption-induced bit-flips in the PCM cells. We have adopted Advanced Encryption Standard (AES)-based counter Mode encryption (CME) due to its low decryption penalty [39, 40]. CME uses a counter (that gets incremented with each write-back of the block), a secret key (stored securely in the processor-side memory controller), and the block address to generate a One-Time Pad (OTP). This OTP is XORed with plain text/cipher text to obtain cipher text/plain text.

Our experiments reveal that many words present in the cache blocks (approaching PCM-based main memory after LLC eviction) are repeated with high frequency over multiple blocks. We term such words as *Popular Words* and store them in a table called Popular Word Table (PWT) for future reference. Encryption of the popular words is skipped during block encryption by maintaining their pointer to the PWT, whereas the non-popular words are encrypted normally. Due to high percentage of popular words in the blocks, encryption of many words are skipped. As a result, compared to the fully encrypted blocks, these Partially Encrypted Blocks (PEB) lead to lesser bit-flips in the PCM cells upon writing. We explicitly dedicate a period for training the PWT to collect the most popular words in the PWT. PWT is dynamically updated during execution to contain the most popular

words for the remaining period of execution. We maintain relevant data structures apart from PWT at low storage overhead to facilitate the partial encryption process.

We compare our technique Pop-Crypt with a baseline that employs full encryption and two state-of-the-art techniques, DEUCE [39] and SECRET [40]. On average, Pop-Crypt reduces bit-flips by 38%; reduces energy consumption by 22%; and improves lifetime by 29%, over baseline, respectively. A detailed discussion of this work is given in Chapter 5.

1.5.4 Exploring Newer Avenues of Bit-flip Reduction in Encrypted NVM Using Compression and Encoding (Contribution 4)

By utilizing the advantages of compression and encoding techniques, bit-flips in encrypted PCMs can be effectively reduced. In this direction, we have proposed two techniques. The working methodology of the techniques is briefly outlined below, whereas a more comprehensive explanation can be found in Chapter 6.

1. CoSeP: Compression and Content-based Selection Procedure to Improve Lifetime of Encrypted Non-Volatile Main Memories

In this technique, we intelligently integrate three compression techniques, FPC, BDI, and COMF, that vary in terms of compression ratio and coverage. FPC shows high coverage but poor compression ratio, whereas BDI offers a fine compression ratio with poor coverage. In contrast, our proposed compression technique, COMF (discussed in Chapter 4), strikes a fine balance between compression ratio and coverage.

An incoming block is compressed independently using FPC, BDI, and COMF, and the sizes of the two smallest compressed blocks are compared. Writing the smallest block after encryption results in the fewest bit-flips in PCM if the sizes differ significantly. This is because the smallest compressed block is much smaller in size. In this case, CoSeP makes a greedy choice by selecting the smallest block for writing in PCM. In contrast, if the sizes of the two smallest blocks differ by only a small margin (i.e., if the blocks are of almost equal sizes), encryption of these similar-sized blocks can lead to significantly different bit-flips, depending on the data content of the compressed blocks. Choosing the smallest block, in this case, is not as wise as it was in the previous case. CoSeP offers the best outcome in such a scenario by computing the bit-flips of the compressed blocks with the old data contents and picking the block that leads to minimum bit-flips after encryption. In that process, CoSeP has to invest some cycles in reading the old data from the main memory to perform bit-flip computation, like the existing read-before-write techniques [42–44] for bit-flip reduction. However, unlike other strategies that perform read operation to determine bit-flip for each block before writing in main memory, CoSeP needs to perform read only when the sizes of the smallest blocks are comparable.

CoSeP has the best overall compression ratio and coverage (0.19 and 95%, respectively) out of the individual compression techniques (0.41 and 84% for FPC, 0.33 and 45% for

BDI, and 0.27 and 75% for COMF). CoSeP achieves 74% and 65% reduction in bit-flips and energy and 69% improvement in PCM lifetime over baseline (encrypted DCW).

2. CADEN : Compression Assisted ADaptive Encoding to improve lifetime of Encrypted Non-Volatile Main Memories : Due to the encryption-induced bit-flip spikes, the traditional encoding techniques [43, 48], developed for non-encrypted memories, fail to achieve significant reduction in bit-flips in the presence of encryption. These encoding techniques can be optimized to perform well in encrypted memories by reducing their encoding granularity. Unfortunately, *fine* encoding granularity can be achieved using more tag bits per block, which substantially increases storage overhead.

Our proposed technique, CADEN applies FNW-based adaptive encoding on the compressed and encrypted data blocks, incurring only a little storage overhead. We apply encoding adaptively on the compressed (generated by our compression technique COMF) and encrypted block by assigning the tag bits to the compressed+encrypted data bits only. Since tag bits point to fewer data bits, the number of data bits represented by one tag bit reduces, resulting in *finer* encoding granularity. Note that we do not need to increase the number of tag bits to achieve *fine* granularity. Furthermore, since COMF produces highly compressed blocks, they reduce exposure to the PCM cells upon writing, reducing bit-flips.

The high compressibility of COMF and *fine* granularity encoding on the compressed blocks enables CADEN to substantially reduce bit-flips in the PCM-based main memory, thereby improving its lifetime. In particular, CADEN shows 52% and 57% reduction in bit-flips and energy consumption and 2.31x improvement in lifetime over baseline.

1.6 Organization of Thesis

The rest of the thesis is organized as follows.

- In chapter 2, the background and prior works related to the contributions of the thesis are discussed.
- We discuss our first contribution in Chapter 3. Here, we propose techniques to reduce costly PCM write-backs in a DRAM-PCM hybrid main memory using a small victim cache associated with the LLC. The victim cache stores critical blocks based on reuse distance. Our proposed techniques further work on the *Block Placement* and *Partitioning* of the victim cache to reduce PCM write-backs further (**Objectives 1, 3, 4, 6(a)**).
- Chapter 4 details our second contribution, where we have proposed a novel compression algorithm called COFAE and an adaptive encoding approach to reduce bit-flips in PCM-based main memory. We also propose an intra-line wear leveling technique to distribute the bit-flips evenly within the PCM cells (**Objectives 1, 2, 3, 4, 6(b)**).

- Chapter 5 presents our third contribution. Here, we have proposed a Partial encryption-based block encoding technique to reduce bit-flips in encrypted PCM-based main memory (**Objectives 1, 3, 4, 5, 6(b)**).
- Chapter 6 illustrates our fourth contribution, where we have proposed new avenues based on compression and encoding techniques to reduce bit-flips in encrypted PCM-based main memory (**Objectives 1, 3, 4, 5, 6(b)**).
- Chapter 7 concludes the thesis.

2

Background and Related Work

As introduced in Chapter 1, the emerging NVMs are widely explored as a potential alternative for the conventional DRAM in main memory. The goal of this thesis is to make NVMs more appropriate as a main memory standard by adopting various ways of write minimization and write distribution. In this chapter, we initially explain the fundamental working principles of the established DRAM-based and emerging NVM-based memory systems. We also discuss the various shortcomings and deployment challenges of the different memory technologies. Additionally, we have discussed various evaluation metrics used in the thesis. Later, we describe the state-of-the-art techniques designed to improve the endurance of NVMs.

2.1 Main Memory Technologies

Below, we provide the working details and drawbacks of different memory technologies used to construct main memories.

2.1.1 Dynamic Random Access Memories (DRAM)

Dynamic Random Access Memory (DRAM) is the most widely used technology to build main memory system. DRAM was invented by Robert Dennard in 1966 at IBM. A storage cell in DRAM is essentially composed of two elements, a *capacitor* and an *access transistor*. The *capacitor* stores the binary values in the form of electrical charge, where the fully charged and discharged states of the capacitor correspond to the binary numbers 1 and 0,

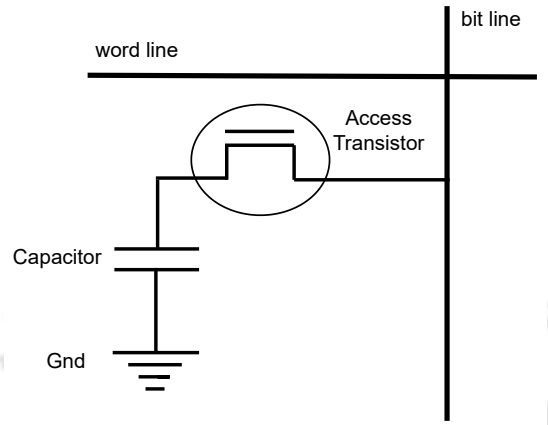


Figure 2.1: *Representational view of a DRAM cell*

respectively. On the other hand, the *access transistor* helps in accessing the bit information stored in the DRAM cell. Two lines, known as word line (W/L) and bit-line (B/L), are used to connect a DRAM cell in a matrix of cells known as sub-arrays that further constitute the DRAM memory banks. The word line and bit line are utilized for the charging/discharging of the capacitor. A schematic diagram of a DRAM cell is shown in Figure 2.1. The read and write operation of a DRAM cell is described below.

Read Operation : To read the bit-value stored in the DRAM cell, voltage is applied to the word line. It causes the current to flow on the bit line. If there is no charge in the capacitor, no current flows.

Write Operation : To write to a DRAM cell, the bit line is set with the appropriate value (1 or 0). The cell capacitor is subsequently charged or discharged (according to the binary value to be stored) by applying a voltage to the word line.

The charge stored in the capacitor is very small to be read reliably. Hence, it is measured by a circuit called *sense amplifier*. The *sense amplifier* is able to pick up on even the slightest changes in charge and relay that information as the corresponding logic level. DRAM reads are destructive, meaning the charge stored in the capacitor flows out during read operation. In other words, a DRAM cell can not represent the bit information accurately after a read operation. As a result, after a read operation, an action known as a *precharge* is carried out to write the bit value back into the DRAM cell. Additionally, the capacitors in DRAM cells have a tendency to leak charge over time, causing the cells to store incorrect bit information. Therefore, DRAM cells need to be *refreshed* periodically to maintain capacitor charge. *Refresh* operations work similar to the read operations and ensure that the cells hold the correct binary value.

DRAM Challenges : The overwhelming demand for main memory with large capacity can not be met due to DRAM's inadequate density. In the effort toward making DRAM denser, it has to undergo a rigorous scaling process. However, scaling DRAM to a low nanometer regime (beyond 10nm) is encountering some unprecedented technological hurdles. Below we highlight the key challenges faced by memory system designers during DRAM scaling.

- DRAM cells must be refreshed periodically due to their tendency to leak charge from the cell capacitors. Unfortunately, the capacitance of the cells drops as their sizes decrease. Additionally, the DRAM cells start to leak more at smaller cell sizes. These two factors contribute to a shortened retention time of the DRAM cells. As a result, DRAM cells need to be refreshed at a faster rate to retain the original information.
- DRAM scaling aggravates the issues related to Variable Retention Time (VRT). VRT refers to the phenomenon where DRAM cells alter between high retention state (less leaky) and low retention (more leaky) over time. It may lead to memory failures during system operations, as some cells can shift to low retention state and start losing bit information.
- The scaling process reduces the spacing between the DRAM cells. As a result, the cells start interfering with each other's operation due to the electromagnetic coupling between the nearby cells. When this disruption exceeds a certain noise margin, the cells start malfunctioning.

2.1.2 Non-volatile Memories (NVM)

NVMs have gained a lot of popularity in recent times as a viable candidate for building main memory systems. Unlike DRAM, NVMs represent data by changing the material characteristics of the cells. Out of many NVMs under research, some NVMs have reached an advanced phase of development, and some are yet to mature. We go over three NVMs that could be used as main memory technologies below: STT-RAM, PCM, and ReRAM.

2.1.2.1 Spin Transfer Torque Random Access Memory (STT-RAM)

STT-RAM is an emerging Magnetoresistive Random Access Memory technology. An STT-RAM cell is composed of a Magnetic Tunnel Junction (MTJ) to store the binary information. A pictorial representation of an STT-RAM cell is shown in Figure 2.2. As shown in Figure 2.2, the MTJ consists of two ferromagnetic layers separated by a thin insulating oxide tunnel barrier layer made up of MgO. One ferromagnetic layer, called *reference* layer, keeps its magnetic direction fixed. In contrast, the other ferromagnetic layer, known as *free* layer, changes its magnetic direction based on the applied spin-polarized current. In an STT-RAM

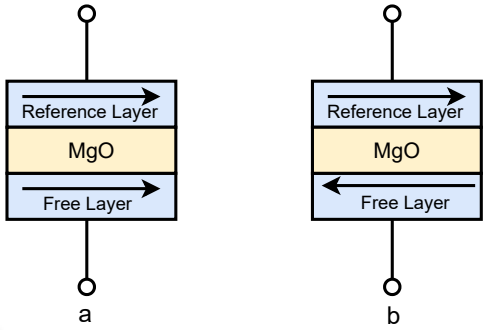


Figure 2.2: *(a) Parallel low resistance, representing logic '0' (b) Anti-parallel high resistance, representing logic '1'*

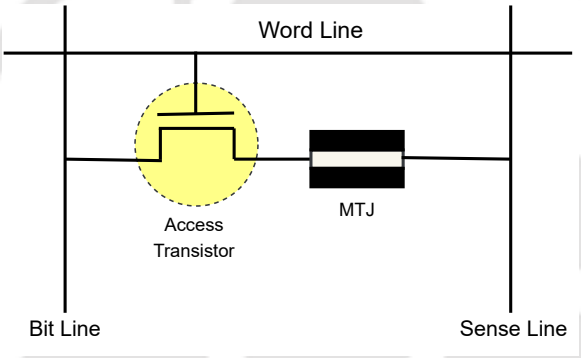


Figure 2.3: *Conceptual view of a STT-RAM cell*

cell, the binary data is stored as the magnetic direction of the *free* layer with respect to the *reference* layer. When the magnetic direction of the *free* layer and *reference* layer are parallel, the MTJ has low resistance and represents a logical 0. In contrast, if their directions are anti-parallel, then the MTJ resistance is high, representing a logical 1.

Similar to a DRAM cell, an STT-RAM cell also consists of an *access transistor* that connects the storage device (MTJ) to its bit line. However, different from a DRAM cell, the other end of the storage device is not connected to the ground; rather gets connected to the sense line. The read and write operations for the STT-RAM cell are described below. The diagram of an STT-RAM cell with the word line, bit line, and sense line is shown in Figure 2.3.

Read Operation : To access the binary value stored in a STT-RAM cell, the *access transistor* of the cell is enabled, and a small voltage is applied between the source line and the bit line. It causes the current to flow across the memory cell, which is compared with a reference current to decide the logical value (1/0).

Write Operation : In order to write a binary value to an STT-RAM cell, a large current is applied to change the magnetic direction of the *free* ferromagnetic layer with respect to

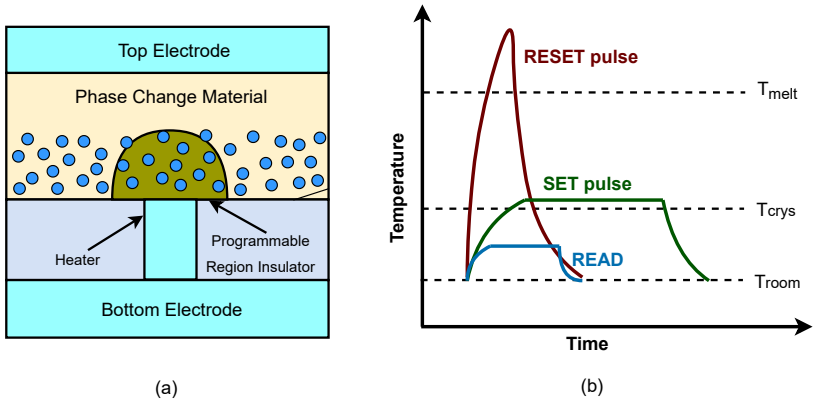


Figure 2.4: *(a) Schematic of a PCM cell (b) Electrical pulses used in read and write (SET/RESET) of a PCM cell*

the *reference* ferromagnetic layer. Depending on the direction of the applied current, the *free* layer becomes parallel or anti-parallel to represent logical 0 or 1 values, respectively.

STT-RAM chips have been commercialized and are available in the market. For example, 4Gbit STT-RAM based MTJ at 90nm technology node is fabricated by Toshiba and SkHynix incorporation [52]. Recently, Intel and Samsung fabricated 7.2M and 8M bit STT-RAM at 22nm and 28nm technology nodes [53, 54], respectively.

2.1.2.2 Phase Change Random Access Memory (PCM)

A PCM cell consists of phase change material, called chalcogenide material, and an *access transistor*. The chalcogenide material, made up of GST (an alloy of Germanium, Antimony, and Tellurium, $Ge_2Sb_2Te_5$), shows two different phases: amorphous and crystalline. The amorphous state shows high resistance and represents the RESET state (logic 0) of the cell, whereas the crystalline state has low resistance and represents the SET state (logic 1) of the cell. The state of the chalcogenide material can be altered between these states by heating the material to different temperatures. The read and write operation of the PCM cell is described below. The schematic diagram of a PCM cell is shown in Figure 2.4(a).

Read Operation : To access the state of a PCM cell, a small voltage is applied across the storage element, GST. Based on the amount of current flowing (sensed using *access transistor* and word line processing), the state of the cell (amorphous/crystalline) that represents the logic value is determined.

Write Operation : To SET a PCM cell, the GST material is heated to more than its melting point (600°C) for a short period and quickly cooled down. It changes the state to amorphous (high resistance), representing logic 0. On the other hand, to RESET a PCM cell, the GST material is heated to a temperature between its crystalline state temperature

(300°C) and melting point (600°C) for a long duration. It changes its state to crystalline state (low resistance), representing logic 1. Figure 2.4(b) shows the waveform of electrical pulses during the Read and Write (SET/RESET) operations of a PCM cell.

The resistance range in the GST is large enough to allow the PCM cells to store more than one bit per cell by dividing the resistance into multiple intermediate levels. It introduces more memory density and results in high memory capacity. These types of PCM cells are known as Multi-Level Cells (MLC). However, MLC PCM has worse latency and energy consumption compared to Single Level Cell (SLC) PCM (PCM cells that we discussed above) as writes are performed in multiple Program and Verify (P&V) iterations to code the bits at the appropriate intermediate resistance level. Due to the high-density benefits, MLC PCM is considered a possible option in the storage layer of the memory hierarchy, whereas SLC PCM is more suitable for main memory.

We consider SLC PCM for our contribution. Hence, PCM necessarily means SLC PCM in the thesis. However, the proposed techniques have the potential to show good results for MLC PCM also as they are applied on the blocks before writing in memory.

PCM has grown a lot in terms of commercialization. For example, Samsung electronics built the 512 Mb PCRAM chip with 266 Mb/s band-width [55], at 90nm node. Later, Samsung fabricated the 8Gb PCRAM chip at 20 nm technology with 40 Mb/s program bandwidth [56]. Intel and STMicroelectronics shipped prototype samples of their first PRAM product, called Alverstone (at 90nm technology node with a capacity of 16MB) to customers [57] in February 2008. Intel's 3D Xpoint is the first commercial PCM chip available in the open market (since 2017) [58]. Optane^{TM} SSD and Optane^{TM} Memory are two memory modules produced by Intel for persistent secondary storage and caching system for HDDs, respectively [59].

2.1.2.3 Resistive Random Access Memory (ReRAM)

ReRAM is built by putting an oxide layer (TiO_2 layer) between an inert electrode (at the top) and an electrochemically active electrode (at the bottom). Figure 2.5 shows a schematic view of a ReRAM cell. On applying a positive voltage to the active electrode, the metal ions move through the oxide layer and reach the inert electrode on the other side. In this state, the equivalent resistance of the cell is taken to be low (1). The low resistance of the cell can be changed to a high-resistance state (0) by applying a positive voltage to the inert electrode. ReRAM has great compatibility with CMOS processes, and therefore, it can be fabricated in the same die where the processor and SRAM-based caches are fabricated. These advantages make ReRAM a strong competitor of SRAM in building cache memories. The read and write operations of a ReRAM cell are described below.

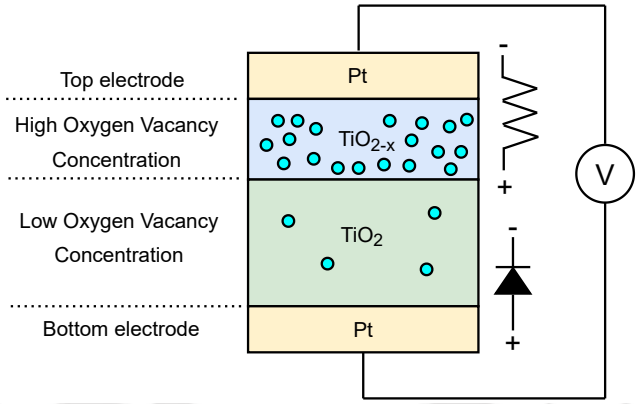


Figure 2.5: *Representational view of a ReRAM cell*

Read Operation : To read the binary value stored in a ReRAM cell, a small voltage is applied across the bit lines. By sensing the generated current, the bit value can be determined.

Write Operation : A large voltage is applied across the bit lines to write a binary bit in a ReRAM cell. To write a 0 bit, a negative bias voltage is used, which generates high resistance ion path by increasing the thickness of the TiO_2 layer, whereas a positive bias voltage is applied to write a 1 bit in the ReRAM cell.

ReRAM has been less commercially utilized compared to STT-RAM and PCM. Fujitsu and Panasonic are working jointly on second generation ReRAM devices [60].

Out of the three memory technologies (STT-RAM, PCM, and ReRAM), PCM and STT-RAM are in matured stage of development and are being deployed at different layers of the memory hierarchy. Table 2.1 [61–66] shows the comparative analysis of the above-discussed NVMs with traditional DRAM. It can be seen from the table that STT-RAM offers higher endurance and lower latency than the other NVMs, making them suitable for building fast on-chip caches, replacing the traditional SRAM-based caches. PCM, on the other hand, shows potential to be used in main memories due to its exceptional density benefits (2-4x compared to DRAM). PCM is the most mature NVM technology with a promising scaling capability. PCM scaling has been verified in a 20 nm device prototype and is projected to scale below 9 nm [21]. Therefore, we have proposed our contribution for PCM-based main memories. However, our techniques are equally adaptable to other NVMs as well.

2.1.2.4 Hybrid Main Memory

Hybrid memory is built using a large NVM memory space and a small DRAM space. It utilizes the low latency of DRAM and high density as well as low leakage power of NVM to construct fast and large-capacity main memory systems. PCM is the commonly used NVM technology in hybrid memories due to its advanced phase of commercialization.

Table 2.1: *Comparative Analysis of different Memory Technologies*

Features	**DRAM**	**STT-RAM**	**PCM**	**ReRAM**
Density	1X	1X	4X	2-4X
Non-volatility	No	Yes	Yes	Yes
Endurance	$>10^{15}$	10^{15}	$>10^{8}$	$>10^{5}$
Read Latency (ns)	10-60	<10	50	<10
Write Latency (ns)	10-60	12.5	40-150	∼10
Dynamic Energy	Medium	High	High	High
Static Energy	Medium	Low	Low	Low
Maturity	Product	Advance development	Advance development	Early development
Retention	<<second	>10 yr	>10 yr	>10 yr

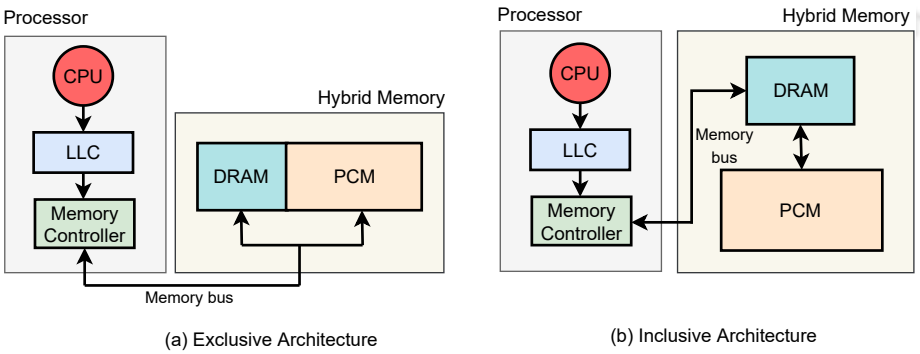


Figure 2.6: *(a) Exclusive (Parallel) organization (b) Inclusive (Hierarchical) organization*

From the design point of view, DRAM-PCM hybrid memory can be divided into two categories, namely 1) *Exclusive Architecture* or *Parallel Architecture* [67, 68] and 2) *Inclusive Architecture* or *Hierarchical Architecture* [69, 70] as shown in Figure 2.6. In *Exclusive (Parallel)* (Figure 2.6(a)) architecture, a small portion of DRAM is maintained with a larger portion of PCM in the same layer of the memory hierarchy. In contrast, *Inclusive (hierarchical)* (Figure 2.6(b)) architecture employs a small layer of DRAM as a cache to the underlying PCM-only main memory.

For applications with low locality of reference, the inclusive architecture cannot provide performance benefits of low latency DRAM cache. Also, the management of DRAM cache in this architecture is an overhead [68]. On the other hand, exclusive architecture fully utilizes both DRAM and NVM memory media and manages to maintain performance even for applications showing low locality of reference. Therefore, we have chosen the exclusive architecture for our work and regard it as the hybrid main memory system. In this thesis, we have explored the exclusive architecture to improve the utilization of hybrid memory.

2.2 Challenges in deploying PCM as the Main Memory Standard

PCM has certain shortcomings that need to be addressed before considering it as a suitable main memory option. Below we discuss the problems associated with deploying PCM in main memory.

2.2.1 Write Related Issues

PCM has asymmetric reads and writes, where the writes are much costlier than the reads in terms of latency and energy. Below we highlight the issues that stem out due to the costly write operations in PCM.

- PCM has higher write latency. Although the writes do not lie in the critical path of execution, they can stall the subsequent critical reads issued to the same memory bank. It can degrade the system performance.

- The write operations involve energy-intensive current injection to the PCM cells. It causes thermal expansion and contraction of the storage elements in the PCM cells and limits their endurance [21]. According to the data in Table 2.1, PCM can withstand around 10^8 writes, whereas DRAM can withstand more than 10^{15} writes. After exceeding the limited write quota, the PCM cells show hard errors where the state of the cell remains permanently stuck at logic zero (RESET/amorphous state) or logic one (SET/Crystalline state) [59].

- PCM cells tend to show non-uniform tolerance to hard errors, where endurance of some cells is more compared to the other PCM cells. This variable write endurance can lead to the early wear out of the more vulnerable cells.
- Programming a PCM cell to high resistance amorphous state (RESET) requires a high current pulse and subsequent cooling of the storage element. The high temperature generated during the RESET procedure can disturb the nearby cells. This type of error is known as *write disturbance* errors [71, 72].

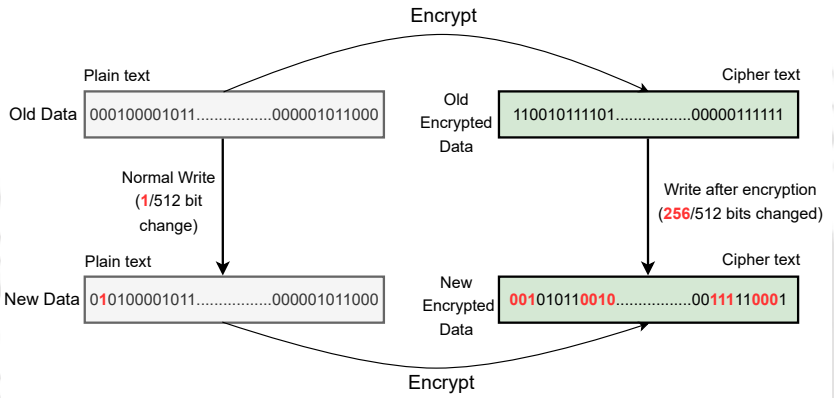


Figure 2.7: *Illustration of Avalanche Effect in Encryption*

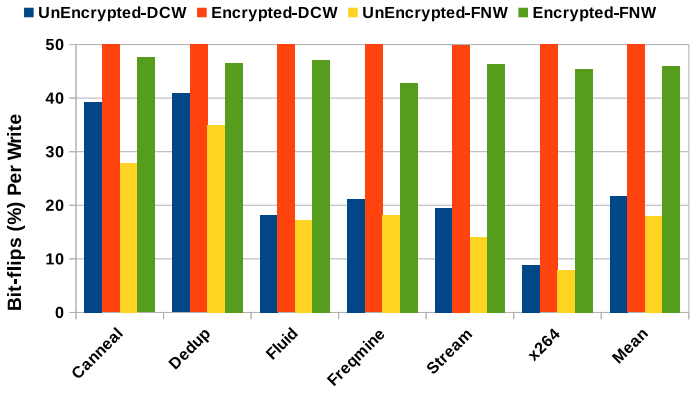


Figure 2.8: *Comparison of bit-flips for Unencrypted Vs Encrypted PCM*

2.2.2 Security Issues : Data Confidentiality-based Attacks

Non-volatility feature of PCM introduces security threats related to *data confidentiality* that were not encountered before. Due to non-volatility, the data in PCM remain persistent for a

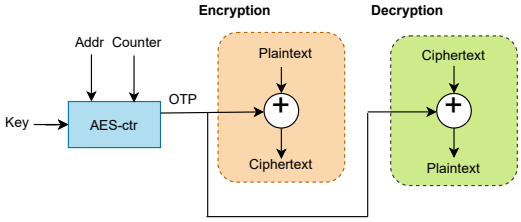


Figure 2.9: *AES-based Counter Mode Encryption*

long duration even after the system is powered off. Therefore, any attacker having physical access to a PCM DIMM can stream out the confidential data without much difficulty.

With its robust obfuscation guarantee, encryption turns out to be an efficient way of guarding PCM data against such malicious attackers. However, security provisioning via encryption is at odds with the PCMs' endurance issue. Due to the high level of randomization imposed by the encryption methods to conceal the data from attackers, PCM endurance degrades further in the presence of encryption. More specifically, even a single bit change in the un-encrypted memory blocks can lead to a large number of bit changes in the encrypted cipher blocks. This effect of encryption algorithms is known as the *Avalanche Effect* [39, 40]. An example of *Avalanche Effect* is shown in Figure 2.7, where a single bit change in the plain text leads to a change of 256 bits in a cache block of 512 bits after encryption. As a result, writing encrypted blocks in PCM causes a dramatic rise in bit-flips because of the high degree of dissimilarity between the old and new encrypted data. In such extreme conditions, the commonly used techniques like DCW [42] and FNW [43] designed to reduce bit-flips in normal condition can not bring fruitful outcome in the presence of encryption. In order to visualize the gravity of this dreadful situation, the graph in Figure 2.8 shows the effects on bit-flips (per write) in PCM-based main memories by DCW and FNW, for un-encrypted and encrypted memories, respectively. In case of un-encrypted PCM, the average bit-flips per write shown by DCW and FNW are 21% and 18%, respectively. In contrast, the bit-flips spike up to 50% and 45%, respectively, for DCW and FNW, when the blocks are encrypted using AES-based encryption. The sudden increase in bit-flips tends to degrade the lifetime of PCM in a severe manner. Also, these costly write activities deteriorate the system performance. As a result, lifetime management while delivering a decent performance is much harder for secured PCM.

Out of many encryption algorithms, AES-based Counter Mode Encryption (AES) works well in the PCM context, primarily due to its low decryption overhead. Below, we discuss the working principle of the AES-based CME in detail. We also discuss the problem associated with CME in Section 2.2.2.2 under *'Counter Overflow Problem'*.

2.2.2.1 Counter Mode Encryption (CME)

CME uses a counter (associated with the cache line), cache line address, and a secret key (stored securely in the processor register) to generate a One Time Pad (OTP). The OTP is XORed with the plaintext data to generate the ciphertext. Similarly, plaintext text can be re-generated by XORing the ciphertext with the OTP. The process of encryption/decryption using AES-based CME is shown in Figure 2.9. The counter associated with a block is incremented on each write-back of the block to maintain *temporal exclusivity* of the counter. *Temporal exclusivity* of counters refers to maintaining unique counter value with time and is essential to avoid counter reuse to protect against dictionary-based attacks [39]. The counters are stored in the main memory, while a few frequently accessed counters are cached in a small on-chip counter cache to reduce decryption penalty.

2.2.2.2 Counter Overflow problem

Small counters are usually preferred to increase hit-rate in the counter cache. However, they overflow quickly after a few write-backs to a cache line. When a counter overflows, the secret global key is changed to avoid the reuse of OTPs, which essentially leads to the re-encryption of the whole memory, as the same global key is kept shared among all the cache lines. It increases bit-flips enormously and causes the system to freeze until the re-encryption process is completed. System freezing has a negative impact on the system performance. We have discussed techniques in the Related Work Section (Section 2.4.2.3) that try to reduce the frequency of counter overflow to improve PCM performance and lifetime.

2.3 Evaluation Metrics

In the thesis, we have used various metrics to assess the effectiveness of our proposed techniques. Below, we discuss the most important metrics used in the thesis.

2.3.1 Lifetime

PCM-based main memories have limited write endurance, i.e., they can withstand only a limited number of writes before wearing out completely. The lifetime of PCM can be defined in two different ways: **Raw Lifetime** and **Error-Tolerant Lifetme**. We define raw lifetime of PCM as the time span until the first breakdown of a byte in such memories. Hence, the lifetime of these memories is inversely proportional to the maximum number of bit-flips to a byte. Raw lifetime can be extended by using error correction techniques [73, 74] at the expense of increased storage overhead and reduced performance. In this thesis, we focus on improving the raw lifetime since it is the base for error-tolerant lifetime also and use the term lifetime in place of raw lifetime for the rest of the thesis. We use the bit-flips in

the cells to determine lifetime of PCM. Since dispersing bit-flips uniformly within memory cells decreases bit-flip pressure within the cells, it contributes to further extending lifetime by lowering the failure rate of the individual cells. The formula for determining lifetime is given by equation 2.1, where N, B are the total number of blocks, number of bytes per block in a memory bank and $W_{blk,b}$ gives the count of writes to b^{th} byte of the blk^{th} block.

$$Lifetime = \frac{1}{\forall_{blk=1}^{N} \forall_{b=1}^{B} W_{blk,b}} \tag{2.1}$$

In this dissertation, we have used raw lifetime which is the basis of error-tolerant lifetime.

2.3.2 Energy Consumption

We follow the energy modeling similar to [41] to calculate the energy consumption in PCM. The energy consumption in PCM is given by equation 2.2, where N_{read} and N_{write} are total read and write operations taking place in the memory, and E_{read} and E_{write} are energy in reading and writing a block in PCM.

$$E_{PCM} = N_{read}E_{read} + N_{write}E_{write} \tag{2.2}$$

E_{read} is the energy involved in reading a 64 byte block from PCM, which is approximately equal to 1.075 nJ [41]. The write energy (E_{write}) involved in writing a block, assuming DCW [42] is enabled, is given by equation 2.3. In equation 2.3, E_{fixed} is the constant amount of energy (E_{fixed}=4.1 nJ per access according to [41]) consumed during write decoding, row selection and data comparison (during DCW), etc.

$$E_{write} = E_{fixed} + E_{read} + E_{bitChange} \tag{2.3}$$

$E_{bitChange}$ is the energy involved in the actual cell writes in PCM. It depends on the nature of flips (0 to 1 or 1 to 0) and is given by the equation 2.4, where $N_{0\to1}$ and $N_{1\to0}$ are the number of bit transitions from 0 to 1 and 1 to 0, respectively and $E_{0\to1}$ and $E_{1\to0}$ are energy consumed in writing a 1 and 0 in the memory cells, respectively. As per [41], the values of $E_{0\to1}$ and $E_{0\to1}$ are 0.0268nJ and 0.013733, respectively.

$$E_{bitChange} = N_{0\to1}E_{0\to1} + N_{1\to0}E_{1\to0} \tag{2.4}$$

2.3.3 Compression Ratio and Coverage

We define the Compression Ratio (CR) as the ratio between the average size of a compressed cache block and the size of the uncompressed cache block. If b_{c1}, b_{c2}....,b_{cn} are the sizes of n compressed blocks, whereas the size of an uncompressed block is denoted by b_{uc}, then the formula for CR is given by the equation 2.5. On the other hand, coverage is the percentage of blocks compressed by compression, which is given by the formula in equation 2.6.

$$CR = \frac{(b_{c1} + b_{c2} + ... + b_{cn})/n}{b_{uc}} = \frac{\sum_{i=1}^{n} b_{ci}}{n \times b_{uc}} \tag{2.5}$$

$$Coverage = \frac{Compressible\ Blocks}{Total\ Blocks} \times 100\% \tag{2.6}$$

2.4 Related Work

In light of the vast potential of NVMs, numerous architectural solutions have been proposed to address their concerns with expensive writes and low write endurance. These works cover the existing techniques for both secure and non-secure memories. We categorize them into two broad divisions, namely *Write Reduction* techniques, and *Wear Leveling* techniques. *Write reduction* techniques reduce the number of writes (at block and bit-level) performed on the memory to improve NVM lifetime. In contrast, the *Wear leveling* techniques distribute the writes around the memory space to equalize the uneven write distribution, thereby minimizing the formation of write hot spots within the memory regions. This thesis mainly focuses on reducing writes to PCM-based main memories. However, we have also proposed a *wear Leveling* technique in conjunction with the proposed write minimization techniques in Chapter 4.

2.4.1 Write Reduction Techniques

Write Reduction techniques operate on *block-level* (coarse granularity) as well *bit-level* (fine granularity) to reduce the write operations in the NVMs. Accordingly, we divide the techniques as *coarse* and *fine* granularity write reduction techniques.

2.4.1.1 Coarse Granularity (Block-level) Write Reduction

These works aim to reduce the write-back traffic of the evicted blocks from the LLC approaching the main memory. Two widely adopted approaches to reduce write-backs are a) Adding a DRAM buffer that acts as a cache to absorb the writes heading towards NVM-based main memories, b) Modifying LLC replacement policy to keep the highly reusable

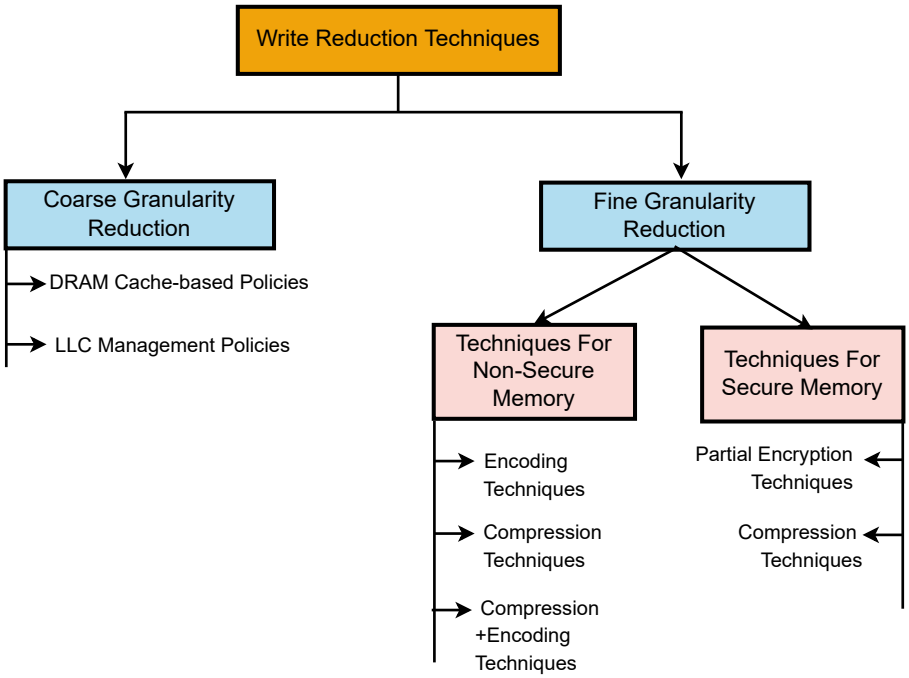


Figure 2.10: *Category Division for the Write Reduction Techniques in NVM*

blocks in LLC for a longer duration. In the following, we will describe the standard state-of-the-art techniques employing both of the aforementioned approaches.

a) Adding DRAM Cache Between LLC and NVM-based Main Memory: This techniques [46, 69, 70, 75, 76] augment a small layer of DRAM acting as a cache to the underlying NVM-based main memory. The DRAM cache is optimized to absorb the write-intensive blocks evicted from LLC. By maintaining the highly reusable and write-intensive blocks in DRAM, it is possible to reduce write-back traffic to NVM and improve its lifetime.

Appeared as one of the initial approaches, Quereshi *et al.*, [69] proposed a series of techniques to utilize the benefits of the DRAM cache by storing the most recently accessed blocks in the DRAM cache. They proposed their techniques for a DRAM-PCM hybrid main memory, where DRAM cache and PCM are organized in page granularity. Their *Lazy Write* policy prevents page-fault induced fill operations in the PCM. When page faults occur, the OS page fault handler allocates the fetched page on the DRAM cache only, skipping its allocation in PCM. It further reduces writes to PCM by evicting only dirty blocks from the DRAM cache. Another technique, called *Line-Level Write Back (LLWB)* proposed in the same paper, minimizes PCM writes by writing at the granularity of a cache line instead of writing the whole page. It keeps dirty bits per block (within a page) to identify the state of a block (clean/dirty).

Part *et al.*, [70] proposed a technique by reserving space (called write buffer) in the

DRAM cache to keep the highly reusable blocks evicted from the DRAM cache. It reduces PCM writes as the reusable blocks evicted from the DRAM cache have more chances to be reused while staying in the write buffer. Based on similar insights, Yoon *et al.*, [75] presented a technique for IoT embedded devices that manages DRAM cache kept as two decoupled buffers, namely *large block fetch* buffer and *self-adaptive filtering* buffer. The *self-adaptive filtering* buffer keeps the highly reusable dirty sub-blocks evicted from the *large block fetch* buffer to give them a second chance. It allows the reusable blocks to stay in the *large block fetch* buffer up to a duration determined using the DRAM cache miss rate and data access pattern.

CAMP [76], optimized for mobile platforms, divides the DRAM space in two regions, namely *page cache* and *dirty block* buffer. The *page cache* is managed at a sub-DRAM page (512 bytes) to improve spatial locality and bandwidth consumption gained through large data chunks. In contrast, the *dirty block* buffer holds the dirty blocks at cache block granularity (64 bytes) to reduce write-backs of the dirty blocks by increasing their occupancy in the DRAM cache.

VAIL [46] is another technique that modifies the replacement policy of the DRAM cache based on the *recency* and *eviction frequency* of the victim blocks. It dynamically monitors the eviction time (that represents recency) and eviction frequency of the victim blocks to predict their reuse possibility. The block that has the least chance of reuse in the future is chosen as the victim. When two blocks have a comparable chance of being reused, their clean/dirty state is examined in order to favorably evict the clean blocks over the dirty ones since they do not result in write-backs to the PCM.

b) Cache (LLC) Management Policy to Reduce NVM Write-backs: These techniques optimize the LLC replacement policies to reduce write-backs to NVM. Some techniques [77–80] are proposed for pure PCM-based main memory, while the others [47, 68, 81] are proposed for the DRAM-PCM-based hybrid main memory with exclusive architecture.

Techniques proposed in [77–80, 82] focus on reducing the write-backs to PCM-based main memory. WADE proposed in [77] tries to keep the frequently written dirty cache blocks in the LLC. It classifies the LLC blocks in two lists, *Frequent write back* and *Non-frequent write back* lists using a frequent write predictor that takes into account both *recency* and *frequency* of the blocks. The sizes of the lists are updated in real-time using a module called *segment predictor* that gives the best sizes of the list to balance between performance and write-back reduction. Zhou *et al.*, proposed two techniques [78] related to write-back aware cache partitioning (WQB) and balancing in write queues to reduce write-backs and improve stall time due to write queue overflow. WCP partitions the cache among competing applications considering reduction in miss rate and write-backs, unlike other partitioning schemes that only considers miss rate. Additionally, the WQB policy uniformly distributes the write requests among many write queues to balance the write traffic. Uniform distribution of the writes throughout the write queues reduces the performance deterioration brought on by

stalls that happen when write queues fill up. Technique proposed by Jie Xu *et al.*, [82] preferentially evicts the cache lines with fewer dirty words. It improves performance by reducing write service time, as write service time is proportional to the number of dirty words in the cache lines.

The techniques proposed in [79, 80] create auxiliary space in the LLC to retain the data blocks that are frequently written. Technique [79] proposed by Bakhshalipour *et al.*, decouples a few physical ways from each set of the LLC to store the frequently written dirty evicted blocks. It uses a predictor to identify the potential re-usability of the blocks in the future. When it is unlikely that a block kept in the decoupled ways will be re-written, it is written back to main memory. The technique called WALL [80] is based on the observation that cache blocks belonging to some sets in LLC face more write-backs, while the other sets remain highly underutilized. It employs the underutilized LLC sets as auxiliary space to hold the frequently written dirty blocks which are evicted from the highly utilized sets.

The techniques [47, 68, 81] are proposed for DRAM-PCM based hybrid main memories. They exploit the observation that the cost associated with eviction from LLC is highly asymmetric for hybrid memories, as the latency and energy involved in write backs to PCM component is significantly larger than the DRAM component of the hybrid main memory. Therefore, decreasing PCM write backs can enhance hybrid memory's performance and longevity. WBAR [47] modifies the insertion and replacement policy of the traditional LRU replacement policy keeping in view the asymmetric read/write disparity of the DRAM and PCM blocks in hybrid memory systems. WBAR promotes and inserts a block at a distinct position in the LRU stack of the corresponding cache set by considering its potential cost of eviction from the set, as opposed to the LRU technique, which attempts to maximize hit rate in the LLC by inserting a newly accessed/incoming block at the MRU position. The potential eviction cost is determined using the type of the blocks (DRAM vs PCM) and its state (clean vs dirty). In particular, considering the higher cost associated during the eviction of the dirty PCM blocks, WBAR tries to keep those blocks for longer duration in LLC. Among the clean blocks, PCM blocks get more preference over DRAM blocks, as reading a PCM block is more latency consuming than a DRAM block if the evicted block is re-referenced again in the upcoming memory reference. Hybrid Aware Partitioning (HAP) [68] introduces a new performance metric called Total Miss Cost Per Kilo Instruction (TMPKI) by taking into account the asymmetric cost involved in DRAM and PCM data misses. It logically partitions the LLC space for DRAM and PCM, giving more share to the PCM blocks to reduce their write backs. However, in order to minimize the TMPKI, HAP adjusts the PCM partition periodically using two thresholds (T_{low}, T_{high}). If PCM partition size is below T_{low}, then performance decreases due to increase in PCM misses. In contrast, if PCM partition size more than T_{high}, performance again decreases due to rise in DRAM misses. HAP maintains a fine balance between the DRAM and PCM partition sizes by dynamically adjusting these two thresholds. It further reduces PCM write backs

using its 2-chance policy that gives a second chance to the dirty victim PCM blocks to reside in the LLC. Since maximizing hit rate may not always result in good performance in hybrid memories, the technique calle MALRU presented by Di Chen*et al.*, [81] uses cache performance based on Average Memory Access Time (AMAT). It partitions the traditional LRU chain of the cache sets in to two partitions, a *reserved partition* and *victim partition*. It keeps the PCM blocks and DRAM blocks with good temporal locality (highly reusable) in the reserved section, which reduces PCM write-backs and more performance is gained due to the reusable DRAM blocks. The boundary between the *victim* and *reserve* partition is dynamically adjusted.

The performance of the techniques based on filtering PCM writes using DRAM cache depends on the utilization of the DRAM cache. However, it becomes difficult for the DRAM cache to maintain high hit rate when the running applications exhibit low temporal locality, leading to increased PCM write-backs and low performance. Also, management of DRAM cache is an overhead in such techniques. On the other hand, the techniques that minimize PCM write-backs by optimizing the LLC replacement policy have to sacrifice some cache space to create auxiliary space [68, 77, 79, 81] to store the evicted dirty blocks. It leads to a reduction in cache capacity and increased cache misses. Furthermore, some techniques [47, 68] do not use any heuristic, whereas some [80, 81] consider the current state of temporal locality to store the dirty evicted blocks in the LLC. However, storing blocks based on such notion may not bring fruitful outcome if the future access pattern of the blocks deviates from the current trend.

2.4.1.2 Fine Granularity (Bit-level) Write Reduction

Bit-flips constitute the fine-granularity write operations in the NVM cells that take place when the fresh blocks are written in NVM. Bit-flips can lead to poor NVM performance and degraded memory lifetime. As discussed in Section 2.2.2, NVM cells face more bit-flips when the data are stored in encrypted form to protect them against *confidentiality* based attacks. The techniques that work on reducing bit-flips can be broadly divided into two groups, *Bit-flip reduction for non-secure memory* and *Bit-flip reduction for secure memory*. Below we highlight the key techniques for each category.

a) Bit-flip Reduction for Non-Secure Memory: Many techniques in this category use data encoding and compression approaches to reduce bit-flips in NVMs. Encoding techniques convert the data bits in the blocks to formats that reduce bit-flips. Compression techniques reduce bit-flips as compressed data blocks affect fewer NVM cells upon writing.

DCW [42] and FNW [43] emerged as early encoding approaches to reduce bit-flips in NVMs. DCW operates by omitting the redundant writes of the unaltered bits. FNW divides the data bits into equal-sized partitions. The partition size is commonly referred to as *encoding granularity*. It inverts the data bits if the number of flipped bits in a partition

is more than half the *encoding granularity*. Else, the bits are written as it is. In this way, FNW restricts the number of bit-flips to half the *encoding granularity*. Each partition has a tag bit assigned to it that indicates whether the bits in that partition are written in the inverted form (tag=1) or as it is (tag=0). FNW reduces bit-flips to a greater extent if the *encoding granularity* becomes finer. However, *encoding granularity* can be made finer by involving more tag bits, which increases the storage overhead. With this insight, Jie Xu *et al.*, developed an encoding technique called READ [44] that dynamically achieves finer *encoding granularity* by assigning the tag bits to the modified words of the block only. Therefore, it does not need to increase tag bits to encode the blocks in finer *encoding granularity*. The authors also suggested another method, named SAE, which is used in conjunction with READ to lessen bit-flips in the tag area by choosing the proper *encoding granularity*, hence minimizing bit-flips throughout the PCM. Captopril [83] proposed by Jalili *et al.*, is based on the observation that there exists a high degree of non-uniformity in the bit-flips experienced by the memory cells when a block is written in PCM. Using a set of hot spot patterns, it divides the blocks into segments and pinpoints the cells that undergo the most bit-flipping within each segment. It keeps the bits corresponding to the hot locations of the partition in inverted form, which is indicated by assigning tag bits to the partitions. FlipMin [48] is a technique based on coset coding [84, 85] that maintains a set of vectors to represent each word within the cache lines. While writing a block, it selects the most appropriate vector for each word that leads to minimum bit-flips in PCM-based main memory. PRES [86] is another coset coding-based method that is based on the finding that coset vectors with more randomness tend to cause fewer bit-flips. Therefore, it maps the data bits (words) into highly random data vectors and subsequently picks up the vector that leads to fewest bit-flips. Suzhen Wu *et al.*, devised an approach called SimiEncode [87] that encodes the words within the cache blocks by XORing them with a mask word, taking use of the data similarity prevalent in the cache lines. The word that occurs most commonly is selected as the mask word. After performing XOR operation of the words with the mask word, the words that are similar to the mask word produce a zero word, which is written in memory with a single tag bit to denote a zero word.

DATACON [88] proposed by Shihao Song *et al.*, is based on the observation that during PCM writing, the SET operation has more latency overhead, whereas the RESET operation involves higher energy consumption. By analyzing the number of SET bits in incoming data and SET/RESET latency-energy trade-off, it writes the data block to an all-zero or all-ones location and stores the translation in a table maintained in the memory controller for future access. It periodically resets unused memory to all-0/all-1 to serve write requests. In order to expedite the address translation process, DATACON caches the frequently used translations.

Most compression-based schemes for bit-flip reduction use FPC [89] and BDI [90] compression techniques to reduce the amount of data written in NVM. FPC compresses the

incoming blocks based on the commonly occurring patterns in the data blocks, which are stored in a pattern table. BDI is based on the observation that the incoming cache blocks to NVM contain many common patterns. These commonly occurring data values follow *low dynamic range i.e.*, they are highly similar and lie with marginal differences within the blocks. In particular, the difference between the words present within the blocks is much smaller in size than the individual words. Based on this, BDI compresses a block by representing it using two common bases (the bases being the first word of the cache line and a zero word) and the small differences between the other words over these bases. Dgien *et al.*, [91] proposed a compression architecture where the data blocks to be written are compressed using an FPC-based compression engine embedded in the NVM module. A read-modify-write protocol is used that updates only the modified bits in the NVM cells. Baek *et al.*, [92] offered a dual-phase compression technique that compresses the blocks using word-level SMA compression and bit-level FPC [89] compression technique. SMA [92] is their proposed technique that compresses blocks at word granularity by removing the second instance of the two adjacently repeating words. In order to uncover often recurring dynamic data patterns in the blocks, DFPC, proposed by Yuncheng Guo *et al.*, [93] takes advantage of the distribution of data values in the words inside the blocks. Unlike the *static patterns* used by FPC, which remain consistent between applications, *dynamic patterns* change from one application to another. DFPC extends the *static pattern* table of FPC by including the *dynamic patterns* and offers scope for high degree compression. The authors proposed an extension of DFPC in [94] to reduce latency and energy in MLC/TLC NVMs. The extension called, Enhanced DFPC (EDFPC) encodes the compressed data blocks into formats that correspond to lower energy/latency states in MLC/TLC NVMs. SimCom [95] is an approximate compression technique used for bit-map data (used for Machine learning , Computer vision) based on the pixel-level similarity in the nearby pixels. It identifies the similar words within the blocks and compresses by representing them as base words and runs (number of similar words).

Some techniques utilize both compression and encoding approaches on the data blocks to minimize bit-flips. Technique [96] proposed by Dan Feng *et al.*, first compares the sizes of the compressed blocks generated by FPC [89] and BDI [90] and then compresses the block using the technique that yields a smaller block size. Finally, depending on the saved space created after compression, it opportunistically applies FlipMin [48] or FNW [43]-based encoding on the compressed data block. FlipMin and FNW have a trade-off in terms of encoding/decoding latency, capacity overhead, and reduction in bit-flips. FlipMin shows better reduction in bit-flips than FNW but incurs more latency and capacity overhead than FNW. In particular, the technique applies FNW when the saved space after compression is low, whereas FlipMin incurring more capacity overhead is applied if saved space is more. The meta-data related to encoding is stored in the saved space to reduce storage overhead. FMR [97] applies FNW [43] on the FPC [89] compressed data blocks to reduce bit-flips.

Based on the realization that FNW can not reduce bit-flip substantially if the hamming distance between new and old data is less than half of partition size, the authors proposed an encoding technique called Mirror-N-Write (MNW) that finds the mirror of the data bits by reversing the order of significance (MSB bits become LSB and vice-versa) of the data bits. It writes the mirror of the data bits if it leads to fewer bit-flips, which is indicated by setting a bit reserved for MNW.

Many encoding techniques [43, 44, 48] discussed above incur high meta-data overhead. These strategies have an inverse connection with storage overhead and efficiency. For example, the ability of FNW and FlipMin to reduce bit-flips becomes high when more tag bits are assigned. Some techniques aim to achieve high reduction in bit-flips while maintaining low storage overhead by assigning the available tag bits to modified data [44] and compressed data [97]. However, the efficacy of these techniques are also limited by data access patterns and efficiency of the compression algorithms. On the other hand, the compression based techniques [91–93] fall behind due to the shortcomings of the existing compression techniques used in NVM compression. FPC and BDI, used by these techniques show a trade off between compression ratio and coverage. FPC shows poor compressibility (high compression ratio) but high coverage whereas, BDI offers high compressibility (low compression ratio) but low coverage. Hence, designing compression algorithms that offer a fine balance between compression ratio and coverage can turn out to be highly essential in improving lifetime of NVMs. At the same time, proposing efficient encoding schemes incurring low storage overhead is equally important.

b) Bit-flip Reduction for Secure Memory: Many techniques have been proposed to reduce the excessive bit-flip surge caused by encryption, which is performed to secure the contents of NVMs against confidentiality based attacks. We divide them into two categories 1) Techniques [39, 40, 98–100] that partially encrypt memory to reduce bit-flips and 2) Techniques [101–103] applying compression to reduce block size.

Partial encryption-based strategies avoid encryption of clean words (within blocks) [39, 40, 99, 104] and inert blocks [98] or eliminate writes to NVM [100, 105]. i-NVMM [98] identifies the working set pages (the pages that are actively accessed by the running applications). Since the working set size is much smaller than the set of inert pages (pages that are not currently being accessed), i-NVMM keeps the working set pages in un-encrypted form while encrypting the inert pages using an encryption engine incorporated in the memory module. BLE [99] divides the incoming cache lines (of size 512 bits) into four partitions of 128 bits in size and assigns a 2-bit local counter to each partition. It performs encryption of the modified partitions during write-backs after incrementing the corresponding counter while keeping the data bits in the un-modified partition in their previous encrypted state. All local counters are reset, and the entire cache line is re-encrypted when a local counter overflows. DEUCE [39] performs partial encryption by encrypting the words inside an incoming cache block that get modified during an epoch interval while avoiding the encryption of the

unmodified words. It uses two virtual counters called the leading counter (LCTR) and the trailing counter (TCTR), derived from the main counter associated with the block. LCTR has the same value as the main counter, whereas TCTR is derived by masking certain bits towards the LSB side of the main counter. The LCTR gets incremented on each write, whereas the TCTR value does not change within the epoch. Within an epoch of writes, the collection of words modified so far are encrypted using LCTR, and the unmodified words remain encrypted in their old values (at the start of the epoch) using the TCTR. At the end of the epoch, LCTR becomes equal to TCTR, and all the words are re-encrypted with the new LCTR. SECRET [40] proposed for MLC NVMs performs partial encryption of the modified words within the incoming cache blocks using per-word local counters. It also prevents the encryption of the zero words, which are tracked using a 1-bit zero flag per word. When a local counter expires, all the local counters of the block are reset, the global counter is incremented by one, and the entire cache line is re-encrypted using the global and local counters. In MLC PCM, programming cells to certain bit patterns consume more energy than others. For example, in 2-bit MLC, encoding cells to 00/11 states are more energy-consuming than 01/10 states. Based on this fact, SECRET performs energy masking on the cipher text data generated after encryption to convert them to low-energy states. Nacre [104] proposed by Tavana *et al.*, performs partial encryption based on the history of data modification in different parts of the cache lines. Nacre divides the cache lines into multiple segments and tracks modification in the segments over the write-backs using segment counters (per segment) and a small amount of meta-data (per block). It encrypts the blocks by XORing the modified segment bits with the OTP generated using the corresponding segment counter. Decryption is performed by XORing the segment bits with their corresponding OTPs, determined using the stored meta-data. Pengfei Zuo *et al.*, proposed a technique called DeWrite [100] that relies on the abundant duplicate data that exists at the line level in the main memory. For each incoming cache block, DeWrite computes a light-weight hash of the block content and looks for a match of this hash value with an already existing block in the memory. On finding a match, it fetches the block and compares it with the incoming data to confirm duplication. The writing of the duplicate blocks is completely canceled, which reduces the spike in bit-flips due to encryption of all the incoming blocks. Silent shredder [105] proposed by Awad *et al.*, reduces writes in NVMs caused by the data shredding operations done by the OS. During data shredding, the OS initializes the contents of each physical memory page to zero before mapping it to a new process, thereby eliminating the possibility of information leakage across the processes. Authors experimentally verified that data shredding operations contribute a large percentage to the overall memory writes. Silent shredder completely eliminates the shredding related writes by making the page contents unintelligible rather than zeroing it by re-purposing the initialization vectors used in CME. When a newly allocated page is read again, a page of zeroes is delivered instead of reading the data from the NVMs. Based on the idea of

applying coset coding like in [48], Longofono *et al.*, proposed a technique called *Virtual Coset Coding* (VCC) [106] that transforms data using a small set of random codewords, called *kernel codes*. The encrypted data block is partitioned into segments, and segments are evaluated against *kernel codes* and their inverses. The final encoded data is created by concatenating either *kernel codes* or their inverse (the one that results in the fewest bit-flips) based on bit-flips.

Some techniques like [101–103] use compression to reduce bit-flips. CryptoComp [101] compresses the data blocks before encryption and applies selective encryption based on the size of the compressed blocks. It applies complete encryption to highly compressed data blocks while encrypting only selected bytes of poorly compressed data blocks depending on the compressed block size. In order to decrease bit-flips in MLC NVMs, CASTLE [102] combines compression and Incomplete Data Mapping (IDM) [107] based encoding. Compression minimizes the effective size of the data that needs to be written, whereas IDM encrypts the cell bits in the states, which results in lesser energy consumption. MORE^2 [103] applies compression and encoding with partial encryption to reduce bit-flips in encrypted PCM. It avoids full-line encryption by encrypting only the modified words of the cache lines. The authors further proposed an encoding scheme called Morphable Selective Encoding (MSE) that compresses the dirty words using FPC [89] while preserving the clean words. In the process of compression, it maintains the data layout of the clean and dirty words.

The techniques discussed above suffer from shortcomings like occasional block re-encryption, high tag storage, overhead related to finding inert/duplicate pages, and poor compressibility of the existing compression schemes. The partial encryption techniques [40, 99, 104] employ per word/segment local counters, which increases meta-data overhead. Furthermore, these techniques need to perform occasional re-encryption of the entire memory line if the local counter corresponding to a word/segment (within the line) overflows. DEUCE [39] also needs occasional re-encryption of the cache line when the epoch interval expires. i-NVMM can not perform well for applications with large working set sizes. In contrast, DeWrite's efficiency reduces when the accuracy of predicting a duplicate block becomes low under diverse workload execution scenarios. Compression-assisted techniques [101–103] fall short due to the poor compression ratio of the underlying FPC algorithm used in these techniques.

2.4.2 Wear Leveling Techniques

Wear leveling techniques distribute the writes evenly across the memory space to even out the write pressure. It reduces the chances of forming write hot spots within memory and helps in improving memory lifetime. Based on the granularity of wear leveling, these techniques can be classified as 1)*Inter-line* wear leveling and 2)*Intra-line* wear leveling. *Inter-line* wear leveling techniques uniformly balance the writes across different physical

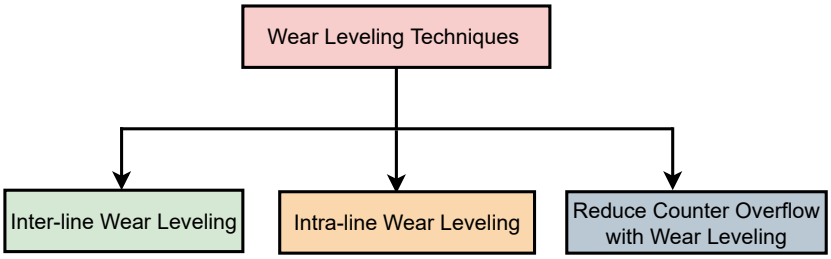


Figure 2.11: *Category Division for the Wear Leveling Techniques in NVM*

memory lines, whereas *Intra-line* wear leveling techniques evenly distribute the bit-flips within the memory lines. Some techniques also deal with reducing encryption overhead in conjunction with wear leveling policies. Figure 2.11 shows the divisions of the techniques related to wear leveling. We discuss techniques related to all these categories below.

2.4.2.1 Inter-line Wear Leveling

Segment Swapping [41] divides the memory into segments and periodically swaps the frequently written (hot) segments with the less written (cold) segments. In order to count the number of writes happened to the segments, each segment is assigned with a counter. Segment size and swapping interval are two important parameters that need careful adjustments. Small segment size increases counter overhead and require time in sorting the segments according to hotness. In contrast, the large counter size increases write overhead during swapping operations. Also, swapping interval must be set properly as frequent swapping imposes high swapping overhead where as infrequent swapping can lead to damage of memory cells due to uneven writing. *Start-Gap* [108] wear leveling approach proposed by Qureshi*et al.*, periodically shifts each memory line to its neighboring memory line with the help of a dummy line, called *Gap* line that contains no useful data. Moreover, it keeps two registers *Start* and *Gap* that point to the first memory line and gap line respectively and help in the translation of logical address to the physical address of the memory lines. *Gap* pointer pointing to the gap line moves to the neighboring memory lines after a certain number of writes to the memory bank. *Start* pointer is incremented when the *Gap* pointer makes a complete tour of all the memory lines in the memory bank. Due to the slow movement of *Gap* pointer, some malicious attacker can damage certain lines by issuing enough writes within the shifting period. Hence, authors move on to propose *Region based Start-Gap* (RBSG) that divides the memory into regions, each operating independently using their own *Start* and *Gap* pointers. RBSG dynamically adjust wear leveling speed based on frequency of writing. In particular, it increases the wear leveling rate for the regions that face heavy writes. In order to prevent the attackers from tracing out useful information about physical locations of the memory lines during wear leveling, Seong *et al.*, proposed a

dynamic mapping scheme [109]. It adds a new level of in-direction by mapping the logical memory address, called *Memory Address* (MA) (obtained from memory controller) to physical locations of the memory blocks called *Remapped Memory Address* (RMA). The mapping scheme XORes the MA with a dynamically generated key K_i. A dedicated circuit called *Security Refresh Controller (SRC)* is incorporated at the memory bank level to perform the mapping at the bank granularity. Self Adaptive Wear Leveling (SAWL) proposed by Huang *et al.*, [110] dynamically adjusts the wear leveling granularity based on the workload access patterns. It stores the address mappings in the main memory while caching a few important mappings in a SRAM-based on-chip cache managed by the memory controller. When the hit rate in the cache becomes fairly low, SAWL merges two/three adjacent memory regions using its *region-merge* operations. On the other hand, if the hit rate in the cache is high and the hits become severely unbalanced within the regions, it splits the regions using *region-split* operations.

2.4.2.2 Intra-line Wear Leveling

Techniques [41, 111] perform byte level rotation to balance the non-uniformity that exists in the cells within the memory lines. *Row shifting* technique presented in [41] rotates memory rows (consisting of many lines) one byte at a time periodically, whereas BLESS presented in [111] shifts the line content byte wise per each write when the line do not contain any errors. When a line faces stuck-at-fault errors, the byte shifting policy is used for error correction than wear leveling. Young *et al.*, [39] proposed Horizontal Wear Leveling (HWL) that operates in conjunction with the inter line *Start-Gap* wear leveling techniques (discussed above). The bits inside the memory lines are rotated periodically by a rotation amount, which is decided using the *Start* and *Gap* pointers. The formula for rotation amount is given by $RotationAmount = Start' \ \% \ Bits\ in\ line$, where $Start'$ equals to $(Start + 1)$ if the *Gap* has already crossed the line, otherwise $Start'$ equals to *Start*. Palangappa *et al.*, [97] proposed an intra-word wear leveling technique called Rotate and Write (RNW) on the compressed data bits produced by FPC. RNW reduces peak bit-flips within the words of the compressed lines. WAFA [112] divides the memory pages into multiple memory units. It allocates the memory units of each page in a rotational manner which distributes the wear uniformly within the pages. Schechter *et al.*, proposed an intra-line wear leveling technique [73] that rotates the data cells within the memory line by random value at random interval of time. Rotation offset is kept in an extra field using extra memory cells. It also keeps one extra cell per memory line that rotates along with the data cells and is used to cover a single stuck-at-fault error.

2.4.2.3 Reducing Counter Overflow in Conjunction with Wear Leveling:

As discussed in Section 2.2.2, counter overflow in CME leads to whole memory re-encryption using a new key, which leads to enormous spike in bit-flips. It also hampers system performance as the normal memory operations are stalled during the entire re-encryption process. Techniques [113, 114] focus on reducing the chances of counter overflow to improve NVM lifetime and performance. Advanced Counter Mode Encryption (ACME) [113] utilizes the disparity in write-backs issued to different memory addresses, where some addresses face more writes compared to the other addresses. As a result, the counter corresponding to the frequently written addresses reach saturation, while other counters remain underutilized. ACME dynamically re-assigns the underutilized counters to the frequently written cache lines when the line is translated from one physical location to another during wear leveling. It leads to uniform growth of all the counters and reduces the chance of counter overflow. COVERT [114] dynamically performs counter re-sizing by using the error correction pointers (ECP) of the memory lines. An ECP contains pointer to a failed memory cell and its correct value. However, as the available ECPs of the frequently written lines get exhausted quickly, dynamic re-sizing is performed in conjunction with the start-gap wear leveling that maps a cache line to multiple memory locations. As a result, size of the fast growing counters can be increased by using the ECPs belonging to newly relocated memory lines, which reduces their chance of overflow. Huang *et al.*, proposed a technique, called RCR (Resetting Counter via Remapping) [115] that resets the counter associated with a line that gets remapped to a different location due to wear leveling. They maintain a region counter to avoid OTP reuse and used start-gap wear leveling. The region counter records the number of complete remapping rounds happened within the region as well as number of remapping to each line. The region counter is concatenated with the line counter to generate OTP during line encryption.

Wear leveling algorithms only redistribute the writes without actually reducing them like the *Write reduction* techniques discussed in Section 2.4.1. Therefore, they can delay the failure of memory cells without avoiding them eventually if heavy writes continue to happen. Furthermore, Table based wear leveling techniques [41, 110] incur the storage overhead of the tables containing logical to physical address mappings. In contrast, algebraic wear leveling [39, 108] use formula to determine the mapping to reduce storage overhead. However, these techniques lack effectiveness since they depend on pointer data structures to decide which line to be moved, rather than simply moving the most write intensive line. For example, *Start-Gap* algorithm always moves the line adjacent to the *Gap* line after certain writes. Hence, write intensive lines may continue to face more writes and tend to wear out quickly. On the other hand, for the intra-line techniques the rotation amount is usually very small. For example, BLESS [111] shifts only at byte granularity and HWL [39] defines rotation based on *Start* pointer that changes very slowly based on the *Gap* pointer. It is because the *Start* pointer gets incremented only when the *Gap* pointer has moved by all the

lines in the memory bank, which can be quite large. Hence, rotation by small strides can lead to uneven distribution to persist even after wear leveling and degrade memory lifetime.

2.5 Summary

In light of the ever-widening gap between processing capability and memory capacity, as well as the proliferation of the modern data-intensive applications, main memory footprint has grown at an unprecedented rate over the years, leading to high dynamic energy consumption. It necessitates re-designing main memory system with newer technology, as the conventional DRAM-based memories are not scalable enough to provide the required capacity and energy efficiency. At this juncture, the emerging non-volatile memories like PCM, STT-RAM and ReRAM appear as a viable alternative to DRAM due to their fascinating features like high density and low leakage power consumption. However, drawbacks like high write latency as well energy and low write endurance hinder their chances of adoption as the mainstream main memory standard.

In this Chapter, we have given details regarding the working principle of different memory technology including DRAM and NVMs, and the challenges associated with using such memory technologies. This thesis focuses on improving the endurance of NVMs by proposing write reduction and write distribution policies. Our contributions also provide strategies to secure NVM data against *data confidentiality*-based attacks using encryption and mitigate the harmful impact of encryption-induced bit-flips. We have discussed the important evaluation metrics used throughout the thesis to measure the effectiveness of our techniques. Finally, we give an extensive literature survey of various techniques used to improve endurance of NVMs.

3

Write Reduction Using Selective Victim Caching

In this chapter, we introduce two new techniques that reduce write-backs to the PCM component of a DRAM-PCM based hybrid main memory using selective victim caching and reuse distance. For our techniques, we associate a small victim cache with the Last Level Cache that helps in retaining the critical DRAM and PCM blocks on chip. Victim cache being a scarce resource, we intend to keep only performance-critical blocks in the victim cache by exploiting the idea of reuse distance. The first technique, Victim Cache Replacement Policy (VCRP), works on the replacement policy of the victim cache by preferential eviction of DRAM blocks over PCM blocks. In contrast, the second technique, Prioritized Partitioning of Victim Cache, logically partitions the Victim Cache (PPVC), giving a smaller share to the DRAM blocks and a relatively larger share to the PCM blocks. Experimental evaluation on full-system simulator shows improvement in system performance and reduction in the number of write-backs to the PCM partition of the main memory compared to the baseline and existing techniques. Additionally, PCM reads and DRAM miss rate are also improved, leading to further performance enhancement.

3.1 Introduction

As discussed earlier, hybrid memories built using traditional DRAM and emerging NVMs exploit the benefits of both memory devices, where DRAM provides fast response due to

its low read/write latency, whereas NVM provides high capacity and low leakage power consumption. However, the performance and durability of the hybrid memories are bottle necked by the existing cache management policies like Least Recently Use (LRU) that are oblivious of the underlying read/write disparity of the DRAM and NVM component of hybrid memories. The traditional cache management techniques can not bring similar performance improvements for hybrid memories mainly due to two reasons: 1) Firstly, evicting a dirty NVM block from the LLC is much costlier than evicting a DRAM block because write latency of NVM is much higher than that of DRAM, 2) Also, the cost in loading a NVM block in LLC from the main memory is costlier than loading a DRAM block since read latency of NVM is higher than DRAM. In order to show the impact of the LRU technique in system performance, an experiment was conducted for a SRAM based LLC in a quad-core environment using LRU replacement policy. Figure 3.1 shows the normalized energy of DRAM-only, hybrid memory (DRAM+PCM), and PCM-only main memory using LRU replacement policy in the LLC. On the other hand, Figure 3.2 shows normalized Instructions Per Cycles (IPC) for DRAM-only, DRAM+PCM hybrid (It is termed as BL1 in experimental evaluation section) and PCM-only main memory system for different PARSEC [116] benchmarks. It is evident from the figures that PCM-only memory performs best and DRAM-only memory performs worst in terms of energy consumption, while the scenario gets completely reversed in case of system performance, in that DRAM being the clear winner. Therefore, we can draw conclusive evidence from the figures that the hybrid memories (DRAM-PCM) cater to the improvements of both system performance and energy consumption in the best possible manner, which otherwise do not go hand in hand. However, traditional LLC management policies like LRU that lack awareness of the underlying disparity in the read-write latency of DRAM and NVMs cannot fully utilize the benefits of the hybrid main memories. Therefore, designing hybrid memory aware LLC management policies that mold in accordance with the requirements of the underlying hybrid main memory has become imperative. It will not only improve performance but also reduce energy consumption in the hybrid memory system.

This chapter proposes two techniques based on the use of a small fully-associative victim cache (VC) placed between the LLC and the DRAM-PCM based hybrid main memory. The main purpose is to reduce write-backs to the PCM section of the main memory without deteriorating the system performance. Also, our techniques work on reducing the PCM reads from main memory, which is also an expensive operation in PCM. Unlike many other existing techniques [47, 68] that sacrifice DRAM miss rate in order to improve the PCM miss rate in the LLC, our techniques not only improve PCM miss rate but also take care of DRAM miss rate by keeping critical DRAM blocks apart from critical PCM blocks in the victim cache. It helps in improving the overall hit rate of the system and promotes further system performance.

Both the techniques exploit the idea of reuse distance in order to identify blocks having

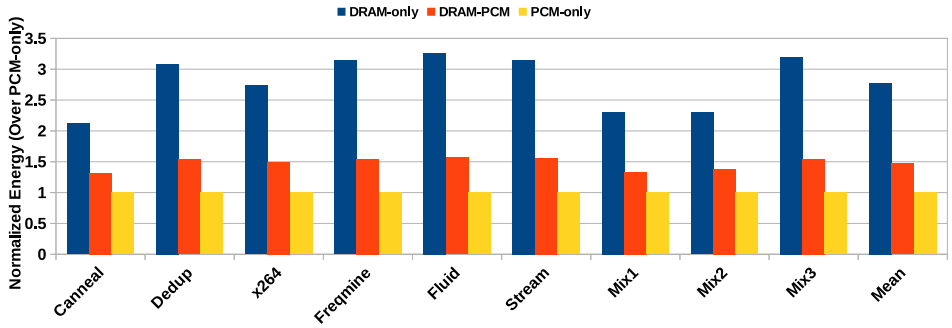


Figure 3.1: *Normalized Energy for DRAM-only, DRAM-PCM Hybrid and PCM-only Memory System (Lower is better)*

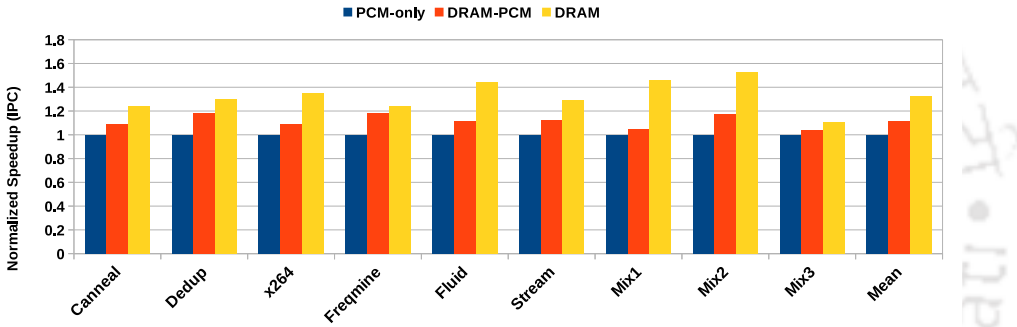


Figure 3.2: *Variation of IPC for PCM-only, DRAM-PCM Hybrid and DRAM-only Memory System (More is better)*

a history of short reuse distance usage during their residency in the LLC. Since short reuse distance is an indicator of high temporal locality, therefore those blocks are treated as candidates for keeping in the VC. Note that our proposed techniques are based on *Frequency of Short Reuse Distance Usage* rather than simply calculating short reuse distance to decide the criticality of the evicted blocks. *Frequency of short reuse distance usage* essentially gives how many times a block was accessed with short reuse distance during its LLC residency. Higher frequency of an evicted block indicates that the block was accessed rigorously within short intervals of time during its LLC residency. Therefore, higher frequency could be taken as a strong precursor that the evicted block has a very high chance of being accessed again in the upcoming memory references. We treat such blocks as critical blocks and consider to store them in the VC. On the other hand, the existing techniques [117–120] tend to keep the blocks having a current short reuse distance (with even less frequency of reuse distance usage) for a longer period in the LLC itself. However, it could be misleading to treat such blocks as critical, since a block showing short reuse distance temporarily does provide very little guarantee that it would follow the same trend in the near future. Therefore, our

techniques are more at par with the temporal locality of the blocks on which the mechanism of caches is based.

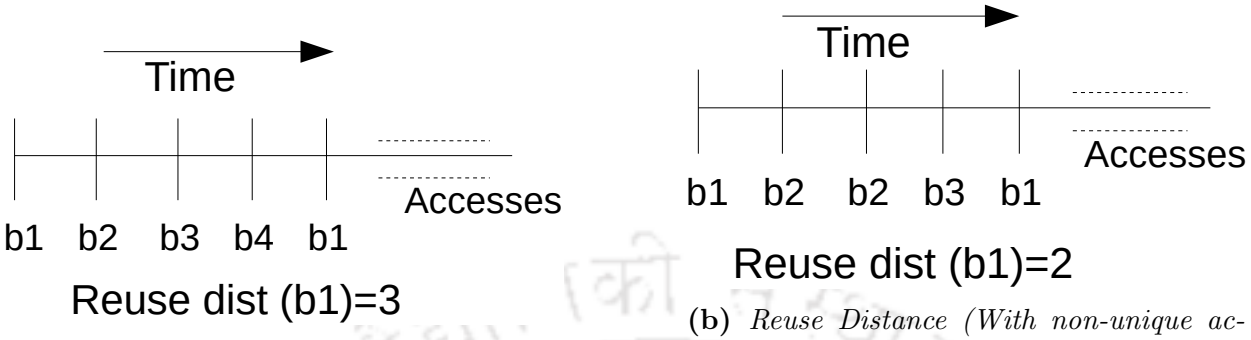
Depending on the type of blocks to be placed (clean blocks vs dirty blocks) in VC, each technique is explained with two variants. Most of the noteworthy works [121–123] using selective victim caching put critical evicted blocks with high reusability in the victim cache in order to reduce the conflict misses, particularly in the L1 Caches. Note that, these techniques were proposed for traditional cache (SRAM-based) and main memory (DRAM-based) systems. On the otherhand, we associate victim cache with the LLC to serve a dual purpose. Firstly, storing of critical blocks evicted from LLC reduces the miss rate due to additional hits gained in the VC, thus improving system performance. Secondly, increase in the hit rate due to VC helps in reducing costly PCM write-backs to the hybrid main memories that is highly crucial in DRAM-PCM hybrid main memory premises. Also, none of these have worked on the victim cache replacement policy and data placement in the victim cache. With a clear contrast with these existing techniques, our policies also work on the replacement as well as the block placement policy of the victim cache that further helps in retaining critical PCM blocks whose evictions are detrimental from the performance and lifetime perspective due to their lazy and energy-intensive write-backs. Note that, the PCM blocks are kept in the VC with more priority compared to the DRAM blocks since the reads and mainly write-backs associated with those blocks are costlier. However, both techniques offer some space to the critical DRAM blocks in the VC so that the system performance does not go down abruptly due to the frequent eviction of these blocks.

3.2 Chapter Overview

3.2.1 Contributions of the Chapter

The main contributions of this chapter are summarized as follows:

- Two techniques based on victim cache are proposed in order to reduce the write-backs of the PCM blocks. Both techniques use the idea of reuse distance to keep critical blocks in the Victim cache.
- The first technique: **V**ictim **C**ache **R**eplacement **P**olicy (**VCRP**) works on the replacement policy of the Victim Cache to retain PCM blocks.
- The second technique: **P**rioritized **P**artitioning of **V**ictim **C**ache (**PPVC**) partitions the VC into two parts, each part dedicated to DRAM and PCM blocks respectively.
- The proposed techniques are evaluated in GEM5 [124] full system simulator integrated with NVMain [125]. The techniques are compared against two baselines (BL1 and BL2) and two existing techniques: WBAR [47] and VAIL [46].



(a) *Reuse Distance (with unique accesses)*

(b) *Reuse Distance (With non-unique accesses)*

Figure 3.3: *Reuse Distance*

3.2.2 Chapter Organization

The rest of the chapter is organized as follows. Background is presented in Section 3.3. Proposed Methodology is discussed in Section 3.4. Section 3.5 illustrates the experimental evaluation, followed by summary in Section 3.6.

3.3 Background

3.3.1 Reuse Distance

Reuse distance [126] of a cache block is the number of distinct accesses that takes place between two consecutive accesses to the block during its residency in the cache. As shown in Figure 3.3, the reuse distance of cache block $b1$ in Figure 3.3a is 3 as there are three unique accesses ($b2$, $b3$, $b4$) between two consecutive accesses to $b1$. However, in Figure 3.3b, the reuse distance of block $b1$ is 2 since the number of unique accesses is 2 ($b2$ and $b3$, and $b2$ is accessed twice). If the cache blocks in a set of a set-associative cache are sorted from MRU to LRU position, the position of a block in the LRU stack denotes its reuse distance. This is because the position of a block in the LRU stack gives the number of unique accesses that has happened to other blocks in the set since the last access to the block. A block accessed with short reuse distance means it is accessed more frequently. A block is said to have a *short reuse distance* if its reuse distance is less than half of the associativity [1] of the cache. The *frequency* of short reuse distance usage is defined as the number of times a block is accessed with short reuse distance during its residency in the cache. Therefore, the frequency of short reuse distance usage can be linked to the locality of blocks in the cache. Higher frequency means that the block is accessed with short reuse

[1] We perform a sensitivity analysis to decide the threshold of short reuse distance as half of the cache associativity (explained in detail in Section 3.5.2.2).

distance more number of times. It essentially indicates high temporal locality of the block. Therefore, the blocks having high frequency of short reuse distance usage are treated as the critical blocks since they have a higher probability of being referred again.

Determining reuse distance (short vs. long reuse distance) to predict future memory access behavior has been in practice for the last few years. [118] proposes a prediction based reuse distance calculation method where they relate a cache line to the load/store instruction accessing the cache line and make prediction according to the previous behavior of the instruction. Reuse Cache [119] by Albericio et al., proposed a decoupled tag/data shared LLC where only data that has shown reuse are kept in the data array. On inserting a cache block for the first time, only the tag array is filled up, while a hit for the block in the tag array indicates more reuse of the block. In that case, the data part of the block is fetched from the main memory and is loaded in the data array.

However, the proposal in this chapter determines reuse distance based on LRU stack, and we rely on the frequency of short reuse distance usage rather than only short reuse distance of a block to declare it as critical. Secondly, the previous works adjust the replacement policy to keep the blocks having short reuse distance for longer duration in the LLC. On the other hand, our policies keep the evicted blocks (from LLC) with a higher frequency of short reuse distance usage (more critical blocks) in the victim cache to increase the hit rate in the VC that helps in improving the system performance. Additionally, storing critical DRAM and PCM blocks in the victim cache reduces costly PCM reads and PCM write-backs while maintaining a fair DRAM miss rate. In a nutshell, our reuse frequency based mechanism turns out to be a great weapon for improving performance as well as lifetime in a DRAM-PCM based hybrid main memory framework.

3.3.2 Victim Cache

Victim Cache initially proposed by Jouppi et al. [127], is a fully associative cache structure that is associated with a cache. Note that, victim cache can be associated with any level of cache. It enhances the system performance by retaining the victim blocks evicted from the associated cache. When a request from upper level cache (ULC) arrives, victim cache is searched in parallel to the LLC it is associated with. In case, the block is found in the victim cache, it is brought to the LLC replacing an invalid entry. If there is no invalid entry available in the LLC, it is swapped with the LRU block of the LLC.

In this Chapter, Victim Cache is used in the LLC (L2 cache) as a buffer in order to store the critical DRAM and PCM blocks, to give them a second chance before evicting completely from the cache hierarchy. Storing PCM blocks will reduce costlier PCM write-backs to the main memory, whereas storing critical DRAM blocks will help to enhance the system performance. Note that, the state-of-the-art [127] victim cache stores all victims evicted from the LLC, whereas the proposal in this chapter only stores the critical blocks

evicted from LLC in the victim cache.

3.4 Proposed Methodology

3.4.1 Architecture

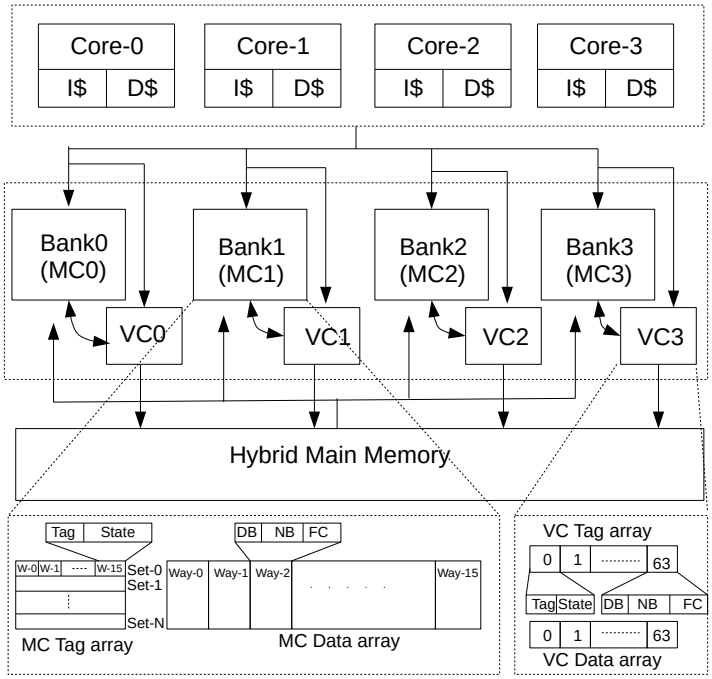


Figure 3.4: *Architecture of LLC associated with VC*

Figure 3.4 shows the representational view of our victim cache-based architecture along with the position of the VC in the memory hierarchy. Unlike many other existing architectures, we associate VC with the SRAM based LLC. We take VC to be a small, fully associative cache structure that sits between LLC bank and the main memory in the memory hierarchy. The search in the VC is done in parallel with the search in the LLC bank. Therefore, VC search does not lie on the critical path of the overall memory search. VC acts as a support for the evicted victim blocks from LLC bank and keeps them in order to give them a second chance before evicting completely from the cache hierarchy. The data array of LLC bank consists of the cache blocks along with a)1-bit (NVM-bit, shown as NB in Figure 3.4 to identify whether the block is a DRAM block or a PCM block. b) a 2-bit saturating counter (Shown as FC in Figure 3.4) to count the number of times the cache

block is accessed with *short reuse distance* during its residency in the LLC bank. When a block is loaded into LLC bank from the main memory, the NVM-bit is set according to the type of the block. If the block is fetched from the DRAM portion of the main memory, NVM-bit is set to 0; otherwise, for blocks belonging to the PCM portion, it is set to 1. Also, the frequency counter is initially set to 0. The frequency counter is incremented if a block gets a hit in the LLC bank, and it has a *short reuse distance*[1]. Cache blocks with higher values of frequency counter are the blocks having a history of *short reuse distance* usage during their LLC residency. Therefore, these blocks are supposed to have high temporal locality and are qualified as potential candidates to be stored in the VC.

VC being a scarce resource owing to its limited capacity, we intend to put only critical blocks in the VC. The criticality of a cache block is defined from two perspectives:

(1) Blocks having *short reuse distance usage* in the recent past are the ones having high temporal locality.

(2) The status (clean Vs dirty) of an evicted block also indicates its vitality to a large extent. Dirty blocks are preferred over clean blocks to store in VC as they cause write-backs to the main memory upon eviction. However, some clean blocks may also have enough temporal locality that might improve the hit rate in a significant manner.

Keeping a strict eye on these two points, we propose two techniques in order to reduce PCM write-backs and improve performance by maximizing hits. Furthermore, to study the behaviour upon placing clean or dirty blocks in the VC, each technique is illustrated under two variants.

3.4.2 Victim Cache Replacement Policy (VCRP)

VCRP works on the replacement policy of VC with an aim to increase the residency of the PCM blocks in VC. Algorithm-1 gives the detailed procedure of what happens when a request R for block b comes to from Upper Level Cache (ULC, L1 cache in our architecture) to LLC bank. The various operations of VCRP are described below:

(1) Hit in LLC: When a request (R) comes for a block (b) to LLC from ULC and hits, its reuse distance is calculated. If it has short reuse distance, its corresponding frequency counter is incremented (line 11). Finally, a copy of the block is returned to the requester ULC (line 12).

(2) Miss in LLC: When a request (R) comes for a block (b) and misses in LLC, the block is searched in VC parallelly. Depending on the availability of the requested block in VC, two cases may arise:

•**Hit in VC**: When the block is found in VC, it is brought to LLC by a) replacing an invalid cache block (line 15). b) If there is no invalid block present in LLC, the LRU block of that

[1]We take the reuse distance of a block to be short if it is less than half of the associativity of the LLC bank.

Algorithm 1: VCRP [VARIANT VCRP_*allD*]

```
1  b: Generic cache block
2  FC: Reuse Frequency Counter
3  NB: NVM bit
4  b_LRU_LLC: LRU block in LLC
5  b_LRU_VC: LRU block in VC
6  b_(LRU−1)_VC: LRU-1 block in VC
7  i: Invalid cache entry
8  repeat
9      for For every request R for block b do
10         if R hits in block b and bϵLLC then
11             if Reusedist(b) < assoc/2 then b.FC++
12             return b to ULC
13         else
14             if R hits in block b and b ϵ VC then
15                 if ∃i ϵ LLC then Move b to i
16                 else
17                     call EvictFromLLC()
18                     Put b in the evicted location
19             else
20                 Fetch data (mem_data) from main memory
21                 if mem_data.type == DRAM then mem_data.NB = 0
22                 else mem_data.NB = 1
23                 if∃i ϵ LLC then Put mem_data in i
24                 else
25                     call EvictFromLLC()
26                     Put mem_data in the evicted location
27 until End of execution
28 Function EvictFromLLC()
29     ifb_LRU_LLC.FC > th2 & isDirty(b_LRU_LLC) then
30     call EntryToVC(b_LRU_LLC)
31     else Evict the block b_LRU_LLC from LLC
32 Function EntryToVC(b′)
33     if∃i ϵ VC then Put b′ in place of i
34     else
35         call VCReplacement()
36         Place b′ in the evicted location
37 Function VCReplacement()
38     ifb_LRU_VC.type == DRAM then Evict b_LRU_VC from VC
39     else
40         ifb_(LRU−1)_VC.type==DRAM then Evict b_(LRU−1)_VC from VC
41         else Evict b_LRU_VC from VC
```

set is evicted and the block found in VC is placed in the evicted location (lines 17, 18).
•**Miss in VC**: Miss in VC triggers a fetch of the requested block from the main memory to the LLC (line 20). The NVM bit is set for the fetched block depending on its type (set 0 for DRAM blocks and 1 for PCM blocks) (lines 21, 22). If the LLC contains an invalid entry, the fetched block is placed there (line 23). If there is no invalid block in the LLC, the LRU block of that set is evicted (line 25) and the fetched block is placed in the evicted location (line 26).

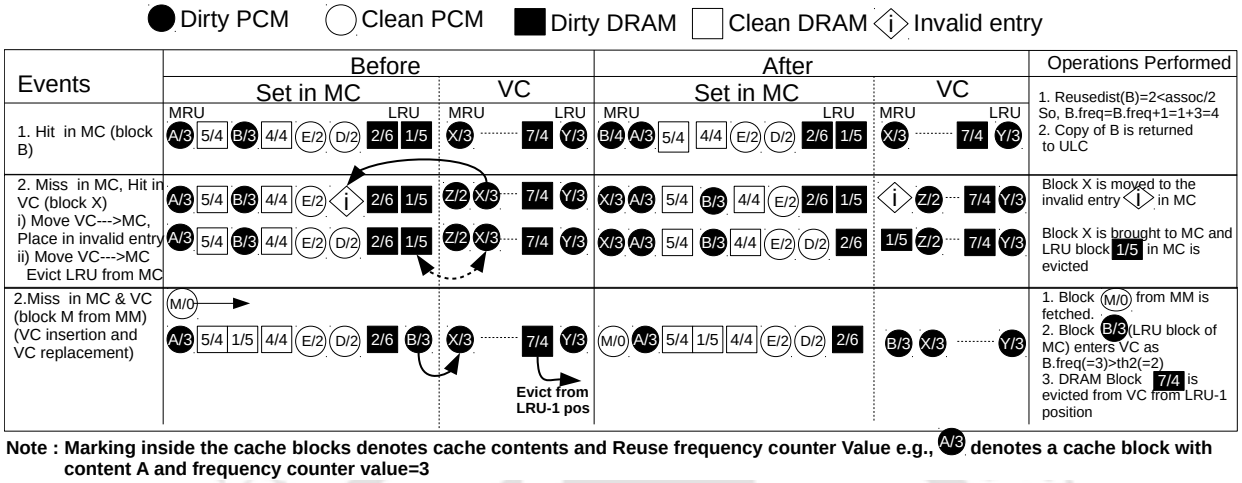
For the evicted block from LLC, depending on the *reuse frequency* counter value and status (clean or dirty) of the block, the block may or may not be stored in the VC, as discussed below:

a. VC insertion Policy: If the *reuse frequency* counter of the evicted block is more than threshold $th2$[1] (indicates the block's high temporal locality), it is considered to be a candidate for placing in VC. Depending on the status (clean or dirty) of the evicted block, we give two variants of VCRP: VCRP_$allD$ and VCRP_$allD_NC$ respectively. For VCRP_$allD$, if the frequency counter of an evicted block is more than $th2$ and the block is dirty, it is placed in the VC (lines 29, 30). On the other hand, for VCRP_$allD_NC$, if the frequency counter value is more than $th2$ and the block is dirty or is a PCM clean block, it is placed in VC.

b. VC replacement policy: Generally, the evicted dirty LRU block is placed in the first invalid entry found in the VC (line 33). In the absence of an invalid entry, another block from the VC needs to be evicted. The block to be evicted must ideally be a DRAM block as more priority is given to keep PCM blocks on the chip. For this, we search the last two positions of the LRU stack of VC. If a DRAM block is found, we evict it, else we evict the LRU block (lines 38-41).

Working example of VCRP: Figure 3.5 shows a working example of VCRP. The LLC is taken as 8-way set associative whereas VC is a 32-way fully associative structure that sits between LLC and the hybrid main memory. The example shows the working of VCRP_$allD$ variant of VCRP. However, the working of VCRP_$allD_NC$ is same as VCRP_$allD$ except it allows the entry of clean, critical PCM blocks into the VC apart from dirty DRAM and PCM blocks. The main operations of VCRP_$allD$ are illustrated under three major events:
•**Hit in LLC:** A request for the block B comes from ULC and hits in the LLC. The reuse distance of B/3 is 2 since it occupies the third position in the LRU stack (sorted from MRU to LRU). Since, the reuse distance is lesser than the half of the associativity of the LLC (assoc/2=4), the frequency counter value of B/3 is incremented by one (becomes 3+1=4). Finally, a copy of the block B/4 is returned to the ULC. The new configuration of the LLC

[1]We did extensive empirical analysis (refer to 3.5.2.3) for different values of $th2$ and found that $th2 = 2$ is the optimal value to decide the criticality of a block for 2-bit *Reuse Frequency* counter.



Note : Marking inside the cache blocks denotes cache contents and Reuse frequency counter Value e.g., A/3 denotes a cache block with content A and frequency counter value=3

Figure 3.5: *A Working Example of the Proposed Methodology: VCRP*

is shown in the figure where **B/4** occupies the MRU position.

•**Miss in LLC, Hit in VC:** A request for block X misses in LLC but hits in VC. Depending on the availability of an invalid entry in LLC, two cases may arise. (i) If there is an invalid entry in LLC (shown in the second row of the Figure 3.5), the block **X/3** is brought from the VC to this invalid entry. A copy of the block **X/3** is returned to the requestor ULC. The configuration of the LLC and VC is shown in the Figure 3.5 after the operation where the block **X/3** occupies the MRU position in the LRU stack. (ii) If there is no invalid entry available in the LLC, the block **X/3** is brought to LLC by evicting LRU block **1/5** of LLC. **1/5** enters VC upon eviction from LLC as it is dirty and has reuse frequency (=5) more than $th2$ (= 2). The blocks **X/3** and **1/5** occupy the MRU positions in the LLC and VC respectively and is cleary shown in the third row of the Figure 3.5.

•**Miss in LLC, Miss in VC:** When the request for a block is missed in both LLC and VC, it triggers a fetch from the main memory. The fourth row of the Figure 3.5 depicts such a situation where the block **M/0** is fetched from the main memory. We take a scenario where no invalid entry is available in both LLC and VC. Therefore, the LRU block **B/3** must be evicted from the LLC in order to make room for the incoming block **M/0**. Since, the reuse frequency counter value (=3) of **B/3** is more than the threshold $th2$ (=2) and it is a dirty block, we consider to store this block in the VC. However, since VC does not have any invalid entry available, we must evict a block from the VC in order to make room for the

block B/3 in the VC. VCRP searches two LRU positions in order to preferably victimize a DRAM block rather than a PCM block. Therefore, the DRAM block 7/4 occupying LRU-1 position in the VC is evicted and written back to the main memory since it is dirty.

3.4.3 Prioritized Partitioning of Victim Cache (PPVC)

The policy VCRP controls the number of PCM and DRAM blocks in the VC based on their reuse distance by concentrating on the replacement policy of VC. Another way to retain a small number of critical DRAM blocks along with a considerably more number of PCM blocks is to partition the VC logically. Here, a small partition is given to the DRAM blocks, and a larger share is dedicated to the PCM blocks. The optimum DRAM: PCM ratio for a better balancing of DRAM and PCM blocks in the VC is experimentally determined and is discussed in the comparative analysis Section 3.5.4.5.

Utility Cache Partitioning (UCP) [128] is one of the key works in cache partitioning. UCP was proposed to partition the shared LLC way-wise among the competing concurrently running applications in different cores in a multi-core environment. The technique offers more cache space to the applications that can reduce their miss rate considerably (non-thrashing applications) by expanding their share of space. However, partitioning of Victim Cache in PPVC has a clear difference with UCP in the sense that it is used in the DRAM-PCM hybrid memory context to provide isolated space in the VC to the critical DRAM and PCM blocks evicted from the LLC bank.

Algorithm-2 gives the procedure of what happens when a request R for a block b comes from ULC to LLC. The various operations of PPVC are discussed below:

1) Hit in LLC: It is same as VCRP and is shown in the lines 12-14 in Algorithm 2.

2) Miss in LLC: If the requested block b is not found in LLC, it is searched in VC. It can result into two cases depending on the availability of the block in VC:

•**Hit in VC:** When the block is present in VC, it is brought to LLC and finally, a copy of it is returned to the requester ULC. Generally, the block is brought to an invalid entry in the LLC (line 17). But, in the absence of such an invalid entry in LLC, the LRU block from LLC is evicted (line 19) and the block found in VC is put in the evicted location in LLC (line 20).

•**Miss in VC:** Miss in VC triggers a fetch from the main memory (line 22). The NVM-bit of the fetched block is set according to its type (0 for DRAM, 1 for PCM) (lines 23, 24). If LLC contains an invalid entry, the fetched block is placed there (line 25). Otherwise, the LRU block of that set is evicted (line 27) and the block fetched from the main memory is kept in the evicted location in LLC (line 28).

For the evicted blocks from LLC, depending on the reuse frequency and type of the block, it may or may not be stored in the VC.

a. VC Insertion Policy: If the frequency counter of the LRU block is more than the

Algorithm 2: PPVC [VARIANT PPVC_*allD*]

```
1  b: Generic Cache block
2  FC: Reuse Frequency Counter
3  NB: NVM bit
4  b_LRU_LLC: LRU block in LLC
5  b_LRU_DRAM: LRU block in DRAM partition in VC
6  b_LRU_PCM: LRU block in PCM partition in VC
7  VC_DRAM: VC Partition for DRAM blocks
8  VC_PCM: VC Partition for PCM blocks
9  i: Invalid cache entry
10 repeat
11     for Each request R for block b do
12         if R hits in block b and bϵLLC then
13             if Reusedist(b) < assoc/2 then b.FC++
14             return b to ULC
15         else
16             if R hits in block b and bϵVC then
17                 if ∃i ϵ LLC then Move b to i
18                 else
19                     call EvictFromLLC()
20                     Put b in the evicted location
21             else
22                 Fetch data (mem_data) from main memory
23                 if mem_data.type == DRAM then mem_data.NB = 0
24                 else mem_data.NB = 1
25                 if ∃i ϵ LLC then Put mem_data in i
26                 else
27                     call EvictFromLLC()
28                     Put mem_data in the evicted location
29 until End of execution
30 Function EvictFromLLC()
31     if b_LRU_LLC.FC > th2 & isDirty(b_LRU_LLC)
32         then call EntryToVC(b_LRU_LLC)
33     else Evict the block from LLC
34 Function EntryToVC(b′)
35     if b′.NB == 0 then
36         if∃i ϵ VC_DRAM then Put b′ in i
37         else Evict b_LRU_DRAM and place b′ in evicted location
38     else
39         if∃i ϵ VC_PCM then Put b′ in i
40         else Evict b_LRU_PCM and place b′ in evicted location
```

threshold $th2$ (line 31), we consider the block to have high temporal locality. Depending on the status of the block (Clean or Dirty) during its LLC residency, we give two variants of PPVC: PPVC_$allD$ and PPVC_$allD_NC$ which are illustrated below:

PPVC_*allD*: If a block has high temporal locality (*reuse frequency* more than $th2$) and it is dirty, it is placed in the VC. If the block is a DRAM block, it is placed in the partition reserved for the DRAM blocks in the VC. Otherwise, it is placed in the partition reserved for the PCM blocks.

PPVC_*allD_NC*: It is same as PPVC_$allD$ except that we allow the entry of PCM clean blocks into VC additionally.

Generally, the evicted block from LLC is stored in an invalid entry in the respective partition in the VC (line 36 for DRAM, line 39 for PCM blocks). However, if the respective partition does not contain any invalid block, a block needs to be evicted from the VC to make room for the incoming block. Here is where the VC replacement policy comes into the picture.

b. VC Replacement Policy: If the incoming block is a DRAM block, we evict the LRU block from the partition reserved for DRAM (line 37). On the other hand, if the incoming block is a PCM block, the LRU block from the partition reserved for the PCM blocks is evicted (line 40).

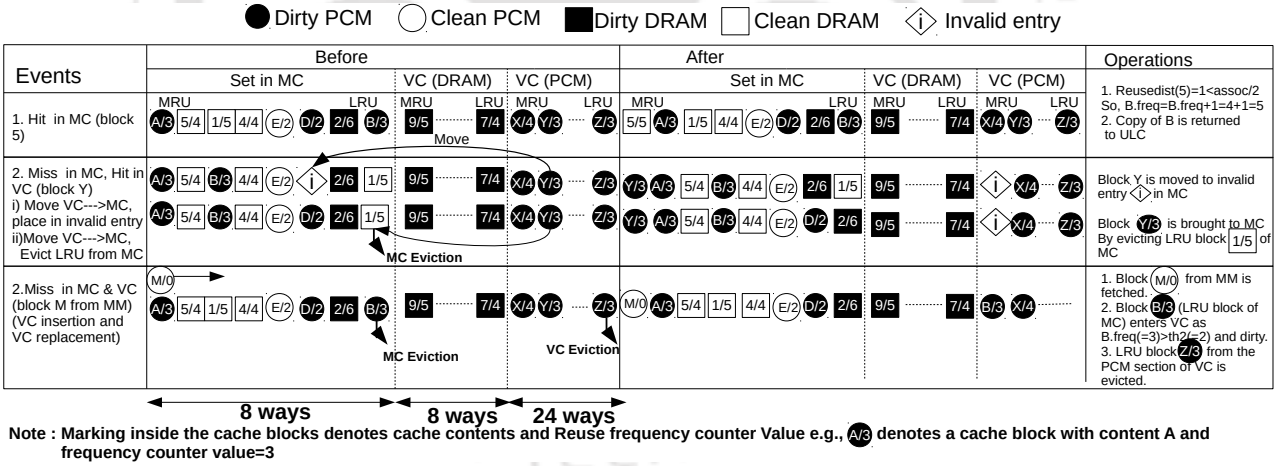


Figure 3.6: *A Working Example of the Proposed Methodology : PPVC*

Working example of PPVC: Figure 3.6 shows a working example of the PPVC_$allD$ variant of the proposed policy PPVC. The working of PPVC_$allD_NC$ is same as PPVC_$allD$ except it allows the entry of clean PCM blocks apart from the dirty DRAM and PCM blocks in the VC. The associativity of the LLC is 8-way set associative whereas VC is a 32-way fully associative structure. VC is partitioned way-wise in order to keep DRAM and PCM blocks separately. The smaller partition consisting of the first 8 ways are reserved for storing

the DRAM blocks and the next 24 ways are reserved for storing the PCM blocks evicted from the LLC. The operations are illustrated below under three main events:

•**Hit in LLC:** A request for the block 5 comes to the LLC from the ULC and hits in the LLC(shown in the first row of the Figure 3.6). Since the reuse distance of the block **5/4** (=1) is lesser than half of the associativity of the LLC (associativity/2=4), its *reuse frequency* counter value is incremented by one (becomes 4+1=5). A copy of the block **5/5** is returned to the requestor ULC. The final configuration of the LLC and VC are shown in the Figure 3.6 where the block **5/5** occupies the MRU position in the LRU stack.

•**Miss in LLC, Hit in VC:** The request for the block **Y/3** misses in LLC but hits in VC. Depending on the availability of an invalid block in LLC two cases may arise: (1) When there is an invalid entry $< i >$ available in the LLC, the block **Y/3** is brought from the VC and placed in the invalid entry. The configuration of LLC and VC after the block is migrated to LLC from VC is shown in the second row of the Figure 3.6. (2) When there is no invalid entry available, the LRU block of LLC is evicted. In the example, LRU block **1/5** of LLC is evicted. This block does not meet the criteria for entering into VC as it is a clean block. Therefore, it is evicted from the LLC without allowing to enter into VC.

•**Miss in LLC, Miss in VC:** A miss in both LLC and VC triggers a main memory fetch of the requested block (block **M/0** shown in the fourth row of the Figure 3.6). We have shown a situation where there is no invalid entry available in LLC as well as VC. Therefore, the LRU block **B/3** must be evicted from the LLC in order to make room for the incoming block **M/0**. Since the block **B/3** is dirty and its *reuse frequency* counter value (=3) is greater than *th*2 (indicating high temporal locality), this block is considered to be placed in the VC. The block **B/3** being a PCM block must be placed in the section reserved for PCM blocks in the VC. However, the LRU block **Z/3** of PCM section in the VC must be evicted from the VC in order to make room for the block **B/3**. The newly entered blocks **M/0** (into LLC) and **B/3** (into VC) occupy the MRU position in LLC and PCM section of the VC which is shown in the Figure 3.6.

Both VCRP and PPVC aim to reduce costly PCM write-backs, as well as PCM reads, yet maintaining a fairly low DRAM miss rate. However, depending on various situations, these techniques have the tendency to bring out unequal profit over one another. PPVC maintains dedicated DRAM and PCM partitions in the victim cache. Therefore, insertion of a PCM block into VC leads to an unavoidable costly eviction of PCM block from the PCM partition when there is no invalid line present in the PCM portion of the VC. VCRP will perform better in this situation as it prefers to evict DRAM blocks than PCM blocks from the VC by scanning the 2-LRU position in the VC. However, there may be a situation, when

Table 3.1: *System Parameters*

Components	Parameters
Processor	2Ghz, Quad-core, X86
L1 Cache	Private, 32 KB SRAM split I/D caches, 2-way set-associative cache, 64 B block, 1cycle latency, LRU, write-back policy
L2 Cache	4 banks, SRAM, 64B block, LRU, write-back policy Size : 4MB (1MB/bank), Associativity : 16-way
Victim Cache	SRAM, 64-way Fully associative, 64 B block size per bank
Main Memory Configuration	DRAM : 1 GB (1 channel), Open page, FRFCFS PCM: 3 GB (3 channel), Open page, FRFCFS-WQF
Memory latency [68]	DRAM :: Read latency : 200 Cycles Write latency : 200 Cycles PCM :: Read latency : 400 Cycles Write latency : 1000 Cycles
Memory Energy (nJ) per block [68]	DRAM :: Read Energy 12.17, Write Energy 14.48 PCM:: Read Energy 10.68, Write Energy 20.75

one class of blocks (DRAM or PCM) get inserted into the VC too aggressively, resulting in the forceful eviction of the critical blocks of the other class by VCRP. However, for such cases, maintaining separate partitions for the DRAM and PCM blocks by PPVC retains a fair collection of both classes of blocks, performing well in such situations.

3.5 Experimental Evaluation

3.5.1 Experimental Setup

The proposed techniques, VCRP and PPVC are implemented on a full system simulator GEM5 [124] integrated with NVMain [125], a cycle accurate memory simulator used for DRAM and NVMs. In order to maintain coherency in both LLC and VC, Ruby module inside GEM5 is used along with MESI CMP based cache controller. The hybrid main memory is modeled using NVMain using DRAM and PCM in 1:3 ratio. Generally, DRAM : PCM hybrid memories are built using small DRAM and large PCM portions to get the high density benefits of PCM without losing the latency benefits offered by DRAM. In order to represent such memory, we have used a small DRAM portion (1 GB) along with a large PCM portion (3 GB) for our experiments. However, a comparison of hybrid memories with DRAM : PCM ratio 1:3 and 2:2 is also provided in the comparative analysis (Section 3.5.4) to show the effects of partition ratio in our proposed techniques. Table 3.1 shows the system parameters used in our simulated system. The initial data allocation in the hybrid memory is the default page allocation process adopted by the Linux OS kernel.

The results are evaluated using PARSEC [116] benchmark suite, which consists of many multi-threaded applications like data mining, animations, multimedia etc. In order to simulate the behavior of multi-programmed benchmarks, we constitute 3 mixes with 4 benchmarks taken from the PARSEC benchmark suite, as shown in Table 3.2. The various energy

Table 3.2: *Benchmarks*

Multi-threaded	Canneal, Dedup, Fluidanimate (fluid), Freqmine (freq), Streamcluster (stream), x264
Multi-programmed	**Mix 1** : Canneal, Dedup, Freq, Stream, **Mix 2** : Canneal, Dedup, fluid, x264 **Mix 3** : freq, fluid, Stream, x264

Table 3.3: *Timing and Energy Parameters for different LLC and VC Configurations*

LLC/ VC	Static Power (mW)	Read Energy (nJ)	Write Energy (nJ)	Read latency (ns)	Write latency (ns)
LLC Size = 1MB	138.7	0.116	0.116	1.874	1.874
LLC Size = 2MB	282.2	0.221	0.221	2.00	2.00
LLC Size = 4MB	554.3	0.330	0.330	2.180	2.180
LLC Size = 8MB	1094.5	0.432	0.432	2.043	2.043
Victim Cache					
VC Size = 16 Entries	2.93	0.007	0.007	0.240	0.240
VC Size= 32 Entries	5.48	0.009	0.009	0.307	0.307
VC Size= 64 Entries	10.44	0.014	0.014	0.416	0.416

and timing parameters of LLC and VC for different sizes are reported in Table 3.3. These values are obtained using NVSIM [129] and CACTI [130]. The results of VCRP and PPVC are compared with two baseline architectures and state-of-the-art techniques WBAR [47] and VAIL [46].

As mentioned in the introduction section, our policies are applicable for exclusive hybrid memories where DRAM and PCM are kept at as part of the same main memory layer. However, in order to show more comprehensiveness, our techniques are compared with another technique VAIL [46] which is applied for inclusive hybrid memories, where a small layer of DRAM is placed hierarchically above the PCM-only main memory. The DRAM layer acts as a cache to the underlying PCM-only main memory. The results are presented in Section 3.5.3.

•**Baseline1 (LLC without VC)[BL1]**: In BL1, a SRAM based LLC without any VC is considered. The replacement policy in the LLC is LRU.

•**Baseline2 (LLC with VC)[BL2]**: In BL2, a SRAM based LLC associated with a fully associative VC is considered. The replacement policy of the LLC and VC is LRU. We do not put any filter on the entry of blocks in the VC. Every block evicted from the LLC is put inside the VC.

•**WBAR**: WBAR[47] is a variant of the traditional LRU replacement policy designed to reduce PCM write-backs in hybrid main memory. It works by changing the insertion and promotion policy of the LRU replacement policy. Instead of inserting or promoting a cache block to the MRU position as in LRU, WBAR places it at a distinct position in a cache set depending on the potential cost incurred if the block is evicted from the LLC.

•**VAIL**: VAIL[46] is proposed for inclusive hybrid memory where a layer of DRAM is placed hierarchically above the PCM-only main memory. The DRAM layer acts as a cache to the PCM-only main memory. It reduces write-backs to the PCM main memory by retaining

Table 3.4: *Translation of VC Entries to Size*

VC Entries	VC Size (KB)
16	1
32	2
64	4
128	8

the blocks with more recency and eviction count.

The proposed policy acronyms used in this chapter are given below.

•**VCRP_*allD*:** Victim Cache Replacement Policy where all dirty critical DRAM and PCM blocks evicted from LLC are put inside VC.

•**VCRP_*allD_NC*:** Same as VCRP_*allD* but also allows the entry of clean, critical PCM blocks into the VC.

•**PPVC_*allD*:** Prioritized Partitioning of the VC where all dirty critical DRAM and PCM blocks evicted from LLC are put inside VC.

•**PPVC_*allD_NC*:** Same as PPVC_*allD* but also allows the entry of clean, critical PCM blocks into the VC.

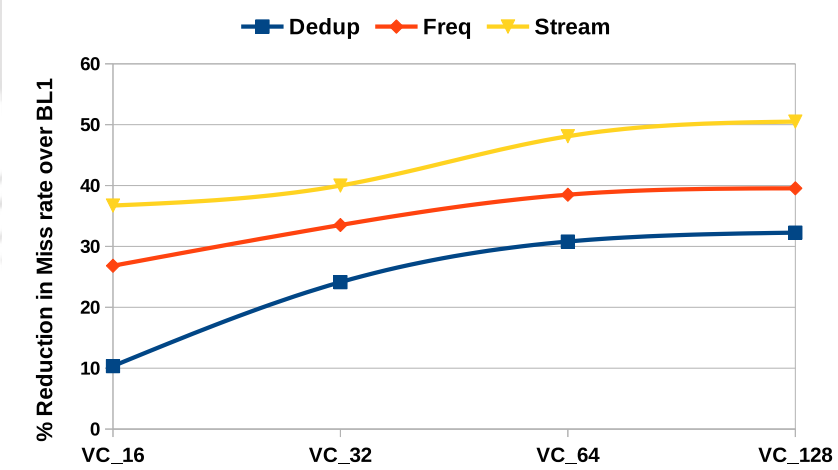


Figure 3.7: *Effect on Miss Rate for Varying VC Sizes (More is better)*

3.5.2 Sensitivity Analysis

Below we give sensitivity analysis on deciding the size of the VC and threshold for short reuse distance.

3.5.2.1 Victim Cache Size

For the size of VC, 64-way fully associative VC is chosen as it balances best the trade-off existing between low search latency associated with a small size VC and high capacity benefits gained from a large size VC. Figure 3.7 shows the percentage reduction in miss rate over BL1 (LLC without VC) for 16, 32, 64 and 128 entry VC. It can be seen that for all the benchmarks, the reduction in miss rate keeps on increasing up to a certain size of the VC (64 entry in our experiments). Increasing VC size beyond this does not have much influence in reducing the miss rate further. Victim cache accommodates more blocks as its size gets larger. However, the varying working set size of the running applications results in the under utilization of the extra space in the VC, as the critical blocks that do not belong to the current working set remain as dead blocks inside the VC. The implication of this phenomenon is the saturation of drop in miss rate beyond a certain threshold of VC size. Therefore, we have taken 64-entries as VC size to carry out all our experiments. Considering the size of a cache block to be 64 bytes, the translation of VC entries to VC size is given in Table 3.4.

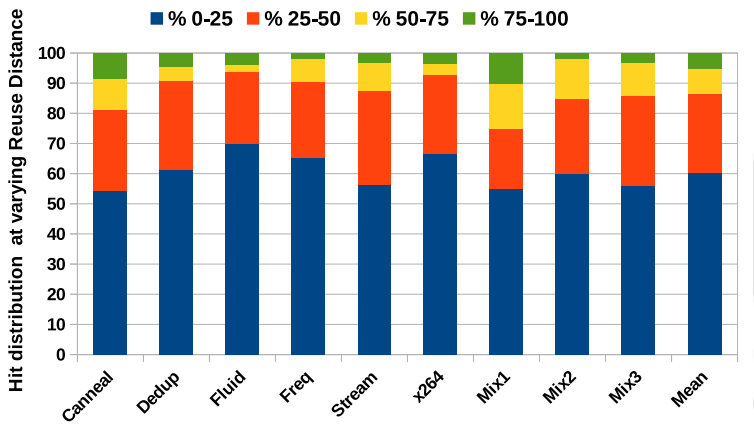


Figure 3.8: *Distribution of Hits at Varying Reuse Distance*

3.5.2.2 Threshold for Reuse Distance

The effectiveness of using victim cache along with LLC increases with the increase in the residence of critical blocks in the victim cache. We perform a sensitivity analysis on the threshold of various reuse distances before declaring a block as critical (having short reuse distance). Figure 3.8 shows the percentage distribution of the number of hits in the LLC obtained at a distance 0-25%, 25-50%, 50-75% and 75-100% of associativity from MRU position across all sets. On average, percentage of hits obtained at distances 0-25%, 25-50%, 50-75% and 75-100% are 60%, 26%, 9% and 5% respectively. It is evident from the

Figure 3.9: *Effect on Reduction in Miss rate and PCM Write-back Reduction for Varying Reuse Distances*

figure that the hits show a decreasing trend with increasing distance from the MRU position in the sets. Most of the hits are obtained at a distance 0-25% and 25-50% of the associativity. Also, Figure 3.9 shows the miss rate and reduction in write-backs (compared to BL1) for distance 25%, 50% and 75% when the LLC is associated with a victim cache. Since the distance from the MRU position essentially denotes the reuse distance, it is evident from the figures that reduction in miss rate and PCM write-backs are commendable up to a reuse distance equal to half of the associativity of the cache. However, they begin to saturate after reuse distance equal to half of the associativity. Therefore, the optimum value of reuse distance to classify a block as a critical should be set to half of the associativity of the LLC. This will help to utilize the victim cache in the most efficient manner by keeping only the most critical blocks evicted from the LLC.

3.5.2.3 Threshold for Criticality

Sensitivity Against Workload Types : Table 3.5 shows the percentage of critical blocks entering VC for th=2, 4 and 8. These blocks correspond to the blocks whose frequency of short reuse distance usage is more than 2, 4 and 8 during their residency in the LLC. As th increases, the percentage of blocks satisfying the threshold condition decreases, meaning less number of critical blocks enters VC for higher th. The impact of changing th is demonstrated through improvement in VC hit rate for varying values of th. The graph in Figure 3.10 shows the increase in hit rate if critical blocks are placed in the VC for different th. The increase in hit rate is shown over the Baseline 2 where all the blocks evicted from LLC are placed in the VC, irrespective of their criticality.

Out of the benchmarks, (dedup, x264) show higher data sharing whereas (canneal, stream) show less data sharing. Accordingly, access frequency of the blocks correspond-

Table 3.5: *Critical blocks entering VC for* th=*2, 4 and 8*

		th=2	*th*=4	*th*=8
High data sharing	Dedup	34%	24%	18%
	x264	36%	24%	6%
Low Data Sharing	Canneal	7%	3%	1.6%
	Stream	20%	12%	8%
Mean		**21%**	**12%**	**6.1%**

Table 3.6: *Effectiveness of Victim Cache over Different LLC Size and Frequency Threshold*

LLC Size	th=2	th=4	th=8
2MB	81%	77%	71%
4MB	42%	37%	35%
8MB	36%	44%	30%

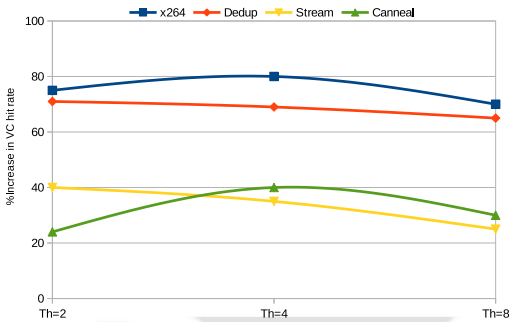


Figure 3.10: *Variation of Hit rate improvement for Th=2, 4 and 8*

ing to dedup and x264 are more compared canneal and stream, which leads to the entry of more critical blocks for dedup and x264. However, It is evident that for all the representative benchmarks, hit rate initially increases for *th*=2 or *th*=4, but decreases for *th*=8. Based on this, we can argue that as the threshold becomes higher, the amount of critical blocks entering into VC reduces. Hence, for higher *th* values, VC space is not utilized properly, leading to low gains in hit rate. Also, the number of counter bits to represent threshold *th*=2, 4, 8 are 2, 3 and 4, respectively i.e., one extra bit is needed for each thresholds. Hence, considering the hit rate in VC and extra counter bit needed for each threshold, we have considered *th*=2 to be the suitable value for our experiments.

Sensitivity Against Memory Hierarchy Parameters: Table 3.6 shows the variation of hit rate in VC for different LLC sizes. Selective victim caching performs well for smaller caches as conflict misses is more for smaller caches and conflict misses lead to critical block eviction. Therefore, VC can provide better support for the critical blocks in case of smaller caches. However, for all sizes of LLC, increase in VC hit rate shows almost the same trend, where it increases initially and gets saturated after $th = 4$.

3.5.3 Results and Analysis

The experiments were done on a quad-core system. We conducted experiments for different configurations of LLC and VC. The system parameters are given in Table 3.1. Out of these configurations, 4-MB, 16-way set-associative LLC is selected for the results. However, results related to various LLC and VC configurations are also reported for better comprehension. In simulations, the extra time taken for the accesses and searching in the victim cache are considered. However, the searching in the VC starts parallelly as soon as the search in the LLC takes place. Therefore, the extra latency taken in the VC search when data is not present in the LLC is of little significance compared to the large latency cost involved in loading the missed block from the main memory in the absence of VC. In particular, 5 cycles are taken during the swapping of blocks between LLC and VC, and 1 cycle extra is taken for searching the block in the VC. The extra cycles are added in simulation in case there is a miss in the LLC. Lastly, the DRAM:PCM VC partition ratio for PPVC is taken as 16:48 for a 64 entry VC. However, comparison of VC partition ratio of 8:56 and 16:48 is presented in Section 3.5.4.

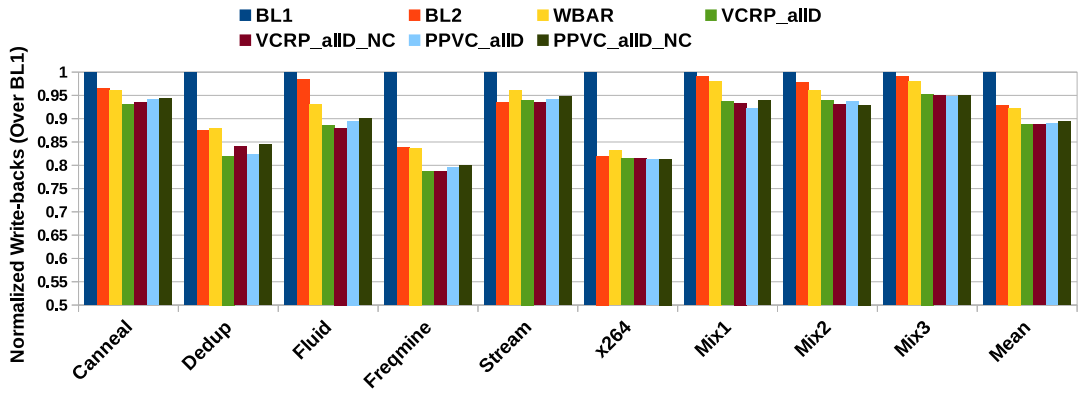


Figure 3.11: *Normalized PCM Write-backs over BL1 (Less is better)*

Effect on PCM write-backs: Figure 3.11 shows the normalized write-backs (over BL1) of VCRP and PPVC. ⟨VCRP_*allD*, VCRP_*allD*_*NC*⟩ reduce write-backs by ⟨ 11.28%,11.26% ⟩,⟨ 5.41%, 5.39% ⟩,⟨ 4.87%,4.86% ⟩ over BL1, BL2 and WBAR respectively whereas ⟨PPVC_*allD*, PPVC_*allD*_*NC*⟩ reduce by ⟨ 11.12%,10.53% ⟩,⟨ 5.24%, 4.6% ⟩ and ⟨ 4.7%,4.1% ⟩. Both VCRP and PPVC outperform WBAR mainly because of the provision of the extra capacity of VC for the PCM blocks. Also, the idea of keeping critical blocks is an added advantage of VCRP and PPVC. VCRP further increases the residency of PCM blocks by its 2-LRU based VC replacement policy, whereas PPVC provides a larger space for the PCM blocks by partitioning the VC in a prioritized manner, both techniques achieving almost 4% improvement over WBAR. The limited capacity of VC is effectively utilized by placing only the critical blocks based on their short reuse distance usage.

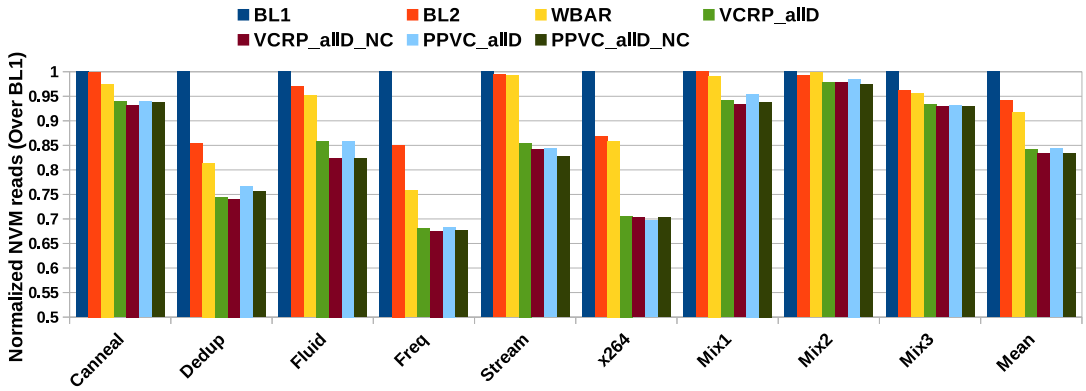


Figure 3.12: *Normalized PCM Reads over BL1 (Less is better)*

Effect on PCM reads: ⟨VCRP_*allD*, VCRP_*allD_NC*⟩ and ⟨ PPVC_*allD*, PPVC_*allD_NC*⟩ reduce the number of PCM reads by ⟨15.8%,16.71%⟩ and ⟨15.58%,16.62%⟩ over BL1 , ⟨10.6%,11.56%⟩ and ⟨10.36%,11.46%⟩ over BL2, ⟨8.24%,9.23%⟩ and ⟨8.0%,9.13%⟩ over WBAR respectively (graph shown in Figure 3.12). WBAR prefers dirty PCM blocks over clean PCM and DRAM blocks for storing in the LLC in order to increase the hit rate (from dirty blocks) and reduce PCM write-backs. However, in our techniques, VC acts as backup storage in order to retain **critical blocks** with high temporal locality. Further, VCRP prioritizes PCM blocks by its 2-LRU VC replacement strategy, and PPVC prioritizes PCM blocks by providing larger partition space to the PCM blocks. Therefore, the probability of PCM read hits increases for the blocks stored in the VC. It helps in reducing the costly PCM reads from the main memory.

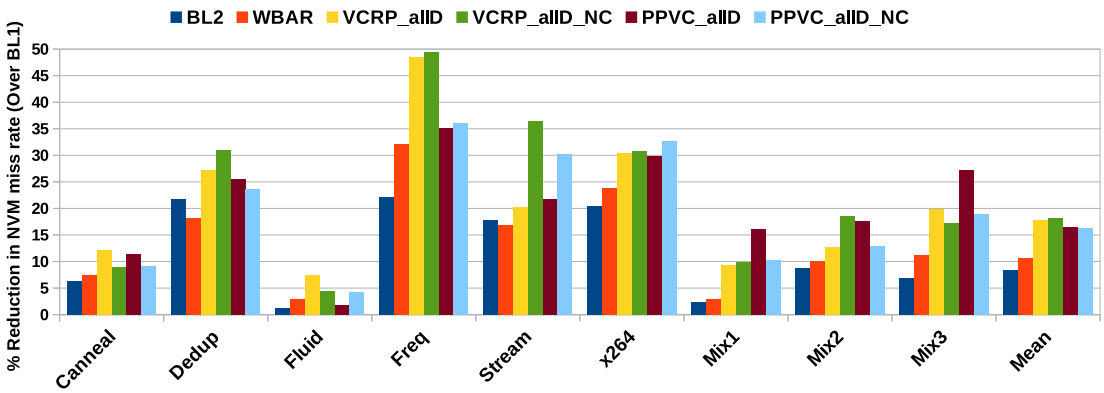


Figure 3.13: *Percentage Reduction in PCM Miss Rate over BL1(More is better)*

Effect on PCM miss rate: Figure 3.13 shows the percentage reduction in PCM miss rate by VCRP (⟨17.76%,18.19%⟩ for ⟨VCRP_*allD*, VCRP_*allD_NC*⟩) and PPVC (⟨16.49%,

16.28%⟩ for ⟨ PPVC_*allD*, PPVC_*allD_NC*⟩) compared to WBAR ⟨10.62%⟩ over BL1. Both VCRP and PPVC outperform WBAR in reducing PCM miss rate, the foremost reason being retainment of critical PCM blocks in the VC. However, VCRP shows better performance compared to PPVC owing to its 2-LRU based VC replacement policy that tends to keep the PCM blocks with higher priority over the DRAM blocks. On the other hand, PPVC does not provide any care through its replacement policy. Therefore, an incoming PCM block into the VC always leads to the eviction of a PCM block from the PCM partition in the VC when there is no invalid cache line available in the VC.

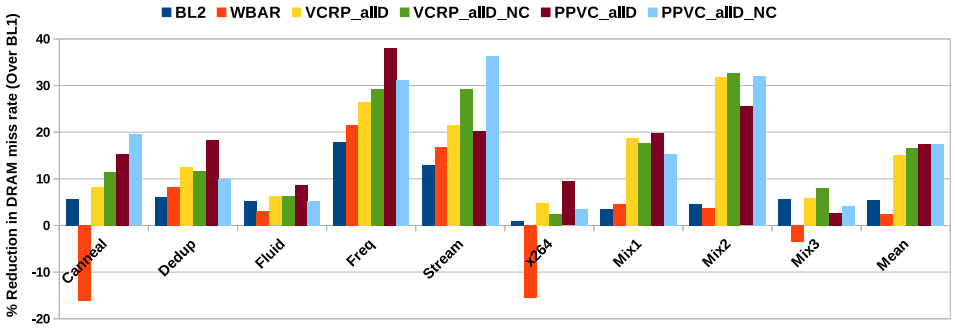


Figure 3.14: *Percentage Reduction in DRAM Miss Rate over BL1 (More is better)*

Effect on DRAM miss rate: Figure 3.14 shows the percentage reduction in DRAM miss rate shown by VCRP (⟨15.11%,16.52%⟩ for ⟨VCRP_*allD*,VCRP_*allD_NC*⟩) and PPVC (⟨17.54%,17.50%⟩ for ⟨PPVC_*allD*,PPVC_*allD_NC*⟩) compared to WBAR ⟨2.53% only⟩ over BL1. Reduction in DRAM miss rate in WBAR is very less (for Canneal, x264, and Mix3 it is negative, meaning DRAM miss rate is more compared to BL1), whereas both the proposed techniques: VCRP and PPVC show a significant reduction in DRAM miss rates. Both these techniques store evicted DRAM blocks with high temporal locality in the VC to give them a second chance. However, the reason that PPVC shows more reduction compared to VCRP is because PPVC maintains a separate reserved section for the DRAM blocks, which increases the residency of the critical DRAM blocks in VC. On the other hand, VCRP preferentially evicts DRAM blocks (by scanning up to 2-LRU positions) in order to give more priority for the retainment of the PCM blocks.

Effect on Performance: Figure 3.15 shows performance improvement by VCRP and PPVC over BL1, BL2 and WBAR. ⟨ VCRP_*allD*, VCRP_*allD_NC* ⟩ show ⟨5.65%,6.01%⟩ improvement over BL1, ⟨3.2%,3.67%⟩ over BL2 and ⟨3.17%,3.54%⟩ over WBAR whereas performance improvement for ⟨PPVC_*allD*, PPVC_*allD_NC*⟩ is ⟨5.62%,5.7%⟩, ⟨3.26%,3,35%⟩ and ⟨3.13%,3.22%⟩ over BL1, BL2 and WBAR respectively.

Referring back to Figure 3.2, where we compared the performance of BL1 (DRAM-PCM hybrid) with PCM only and DRAM only memory systems, the graph in the Figure 3.15 shows that VCRP and PPVC reduce the gap between DRAM-only memory and BL2

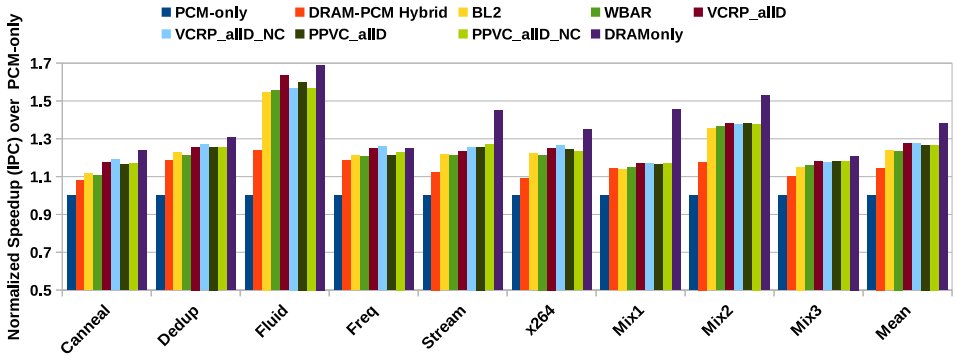


Figure 3.15: *Performance Improvement over PCM-only (More is better)*

(DRAM-PCM hybrid) to a greater extent compared to WBAR. WBAR bears a loss of 10.31% performance over DRAM-only, whereas loss in performance in VCRP and PPVC are ⟨7.71%,7.4%⟩ and ⟨7.82%,7.74% ⟩ (for both variants of PPVC and VCRP) respectively. In other words, our policies give around 3% improvement in performance over WBAR.

VCRP and PPVC allow the entry of critical evicted blocks in VC. VCRP further improves performance by increasing the residency of PCM blocks over DRAM blocks by working on the eviction strategy (2-LRU based eviction) of VC. VCRP_*allD_NC* performs better than VCRP_*allD* due to additional hits gained in VC from the clean PCM blocks. PPVC differs from VCRP in the block placement strategy in VC. It saves costlier PCM write-backs by retaining more PCM blocks due to prioritized partitioning of the VC, thus improving performance.

As discussed in the results for PCM miss rate and DRAM miss rate, VCRP works more effectively in reducing PCM miss rate, while PPVC reduces DRAM miss rate more compared to VCRP. In other words, the effect of system performance gain in VCRP and PPVC is primarily due to giving the retainment priority to critical PCM blocks (for VCRP) and critical DRAM blocks (for PPVC), respectively. The overall effect in performance improvement is almost same for the two proposed techniques.

Comparison Across Benchmarks: PARSEC benchmarks show wide variation in terms of data sharing behavior, temporal locality, and off-chip bandwidth requirements [116, 131, 132]. Therefore, the improvements obtained by our techniques are different depending on the behavior of the running benchmarks. Dedup, Freqmine, and x264 come under high data sharing benchmarks where cache blocks are shared heavily among different threads running across the cores. Therefore, cache blocks belonging to these benchmarks show high reusability or possess short reuse distance. Therefore, storing such critical blocks in VC bring more improvements, as it is reflected in the improvements of PCM reads/write-backs as well as performance improvements. (On average, (Dedup, freq, x264) shows improvements of (18%, 21%, 19%) in PCM write-backs, (25%,32%,29%) in PCM reads and (7%, 15%,

17%) in performance (over BL1) for our techniques). On the other hand, Canneal and Streamcluster are streaming applications where data reuse is very less. Also, they have high off-chip bandwidth demand for load, store, and writeback operations. Therefore, storing blocks of these benchmarks in VC brings less improvements compared to Dedup, freq and x264 (On average, (Canneal, Streamcluster) show improvements of (7%,6%) in PCM write-backs, (6%,12%) in PCM reads and (7%, 9%) in performance (over BL1)).

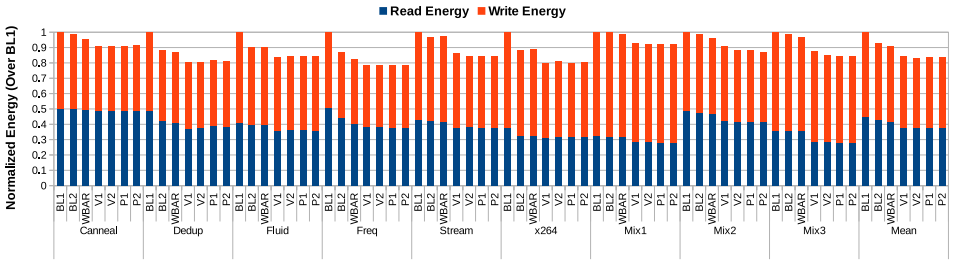


Figure 3.16: *Normalized Energy over BL1 (Lesser is better) (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

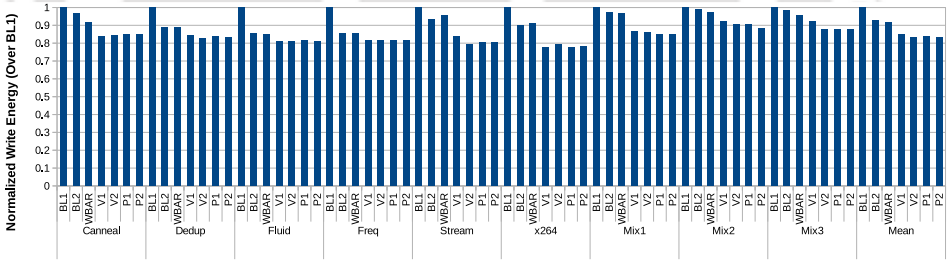


Figure 3.17: *Normalized Write Energy over BL1 (Lesser is better) (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

Effect on Energy: Energy consumption in hybrid memory mainly consists of the sum of read energy and write energy. The formula for calculating energy is given by equation 3.1. Since writes consume more energy as compared to reads in case of PCMs, therefore techniques that reduce the costly PCM write-backs can contribute immensely to the reduction of overall energy consumption in hybrid memory. The prime objective of both of our proposed techniques is to reduce PCM write-backs that help in reducing energy consumption. Additionally, caching of critical DRAM and PCM blocks in the victim cache also reduces

DRAM and PCM reads, which assist further in the reduction of energy consumption.

$$\begin{aligned} Energy = \#DRAMRead\ Accesses \times DRAMRead\ Energy(Per\ Access) + \\ \#PCMRead\ Accesses \times PCMRead\ Energy(Per\ Access) + \\ \#DRAMWrite\ accesses \times DRAMWrite\ Energy(Per\ Access) + \\ \#PCMWrite\ accesses \times PCMWrite\ Energy(Per\ Access) \end{aligned} \tag{3.1}$$

All the variants of the proposed techniques: VCRP and PPVC reduce PCM write-backs as well as PCM reads significantly. As a result, energy consumption is fairly less in the proposed techniques. Figure 3.16 shows reduction in energy for ⟨VCRP_*allD*, VCRP_*allD_NC*⟩ by ⟨11.78%,11.68%⟩, ⟨5.95%,5.86%⟩, ⟨4.8%,4.7%⟩ over BL1, BL2 and WBAR respectively where as reduction in energy by ⟨ PPVC_*allD*, PPVC_*allD_NC* ⟩ is ⟨11.65%,11.4%⟩, ⟨5.82%,5.55%⟩ and ⟨4.66%,4.4%⟩ respectively. In particular, the reduction in write energy for ⟨VCRP_*allD*, VCRP_*allD_NC*⟩ is ⟨17.77%,18.42%⟩, ⟨10.6%,11.3%⟩ and ⟨9.31%,9.34%⟩ over BL1, BL2 and WBAR respectively whereas reduction in write energy in case of ⟨ PPVC_*allD*, PPVC_*allD_NC*⟩ is ⟨18.44%,18.24%⟩, ⟨11.34%,11.12%⟩ and ⟨9.37%,9.3%⟩ respectively. The graph in the Figure 3.17 clearly shows the trend in the reduction of write energy across different benchmarks for BL1, BL2, WBAR and our proposed techniques.

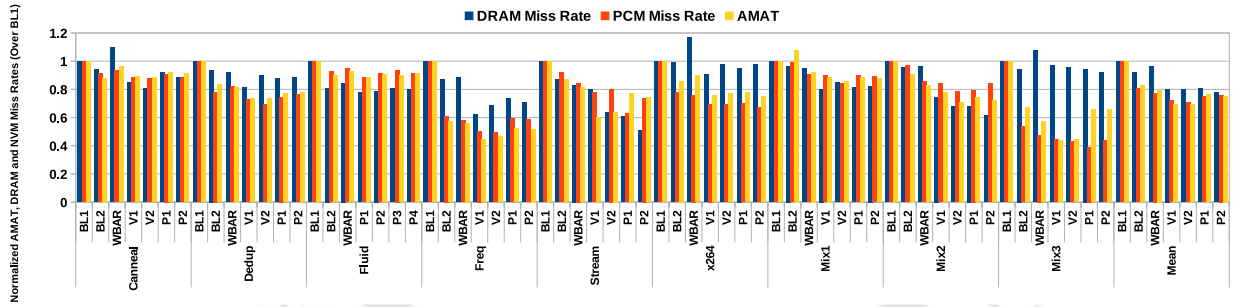


Figure 3.18: *Normalized AMAT over BL1 (Lesser is better); (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

Effect on AMAT: Average Memory Access Time (AMAT) gives the average estimation of time taken to deliver the requested data items from the memory hierarchy. An improved AMAT (lower is better) indicates an agile memory system that assists in improving the system performance. Considering the non-uniform load latencies of DRAM and PCM in DRAM-PCM hybrid memories, there lie ample opportunities to work on this asymmetry to improve AMAT in such systems. The formula for calculating AMAT in a DRAM-PCM hybrid memory system is given by equation 3.2.

$$\begin{aligned} AMAT = HitTime(L1) + MissRate(L1)[HitTime(L2) + DRAMMissRate(L2) \\ \times DRAMMissPenalty + PCMMissRate(L2) \times PCMMissPenalty] \end{aligned} \tag{3.2}$$

Fortunately, both of our proposed policies, VCRP and PPVC work on reducing both DRAM and PCM miss rate in the LLC by keeping critical blocks in the associated VC. However, the existing technique WBAR that primarily focuses on reducing PCM write-backs does not pay much attention to improve the DRAM miss rate. This simple yet highly effective improvisation of reducing DRAM miss rate gives our techniques an extra edge in improving AMAT more than WBAR. The variation of AMAT along with DRAM miss rate and PCM miss rate is shown in Figure 3.18 for baselines (BL1, BL2), WBAR, and our proposed techniques. On average, percentage improvement in AMAT shown by WBAR over BL1 is 17% only whereas our proposed techniques: ⟨VCRP_*allD*, VCRP_*allD_NC*⟩ and ⟨PPVC_*allD*, PPVC_*allD_NC*⟩ show a considerable improvement of ⟨31%, 30%⟩ and ⟨24%, 25%⟩ over BL1 respectively. The reduction in DRAM miss rate for WBAR (over BL1) is only 2.53% whereas reduction in DRAM miss rate by ⟨VCRP_*allD*, VCRP_*allD_NC*⟩ and ⟨PPVC_*allD*, PPVC_*allD_NC*⟩ are ⟨15.11%, 16.52%⟩ and ⟨17.54%, 17.5%⟩ respectively. Similarly, reduction in PCM miss rate by WBAR is 10.62% whereas whereas reduction in PCM miss rate by ⟨VCRP_*allD*, VCRP_*allD_NC*⟩, ⟨PPVC_*allD*, PPVC_*allD_NC*⟩ are ⟨17.76%, 18%⟩, ⟨16.5%, 16.28%⟩ respectively.

Comparison of AMAT of VCRP and PPVC : VCRP performs better than PPVC in reducing AMAT because of the difference in its eviction policy. In the case of PPVC, when a PCM block enters the VC, it must be entered into the respective PCM partition in the VC. It results in the eviction of the LRU block in the PCM partition from the VC in order to make room for the incoming PCM block. It increases the costly PCM write-backs, which results in increase in AMAT in the case of PPVC. On the other hand, in the case of VCRP, when a PCM block enters into the VC, eviction of a PCM block from the VC in not mandatory since VCRP gives priority to the eviction of DRAM blocks from the VC upon replacement by scanning the 2-LRU position in the VC.

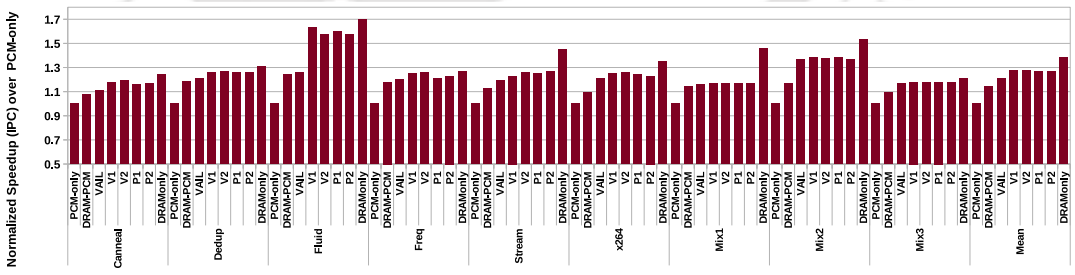


Figure 3.19: *Normalized IPC over PCM-only (Lesser is better); (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

Comparison With Another Existing Policy : VAIL [46] proposed by Fang Zhou *et al.*, has similar objectives of reducing costlier PCM write-backs by modifying DRAM cache replacement policy in a DRAM-PCM inclusive architecture. It uses a small DRAM cache lying between the LLC and the PCM-only main memory that increases hit rate in the DRAM

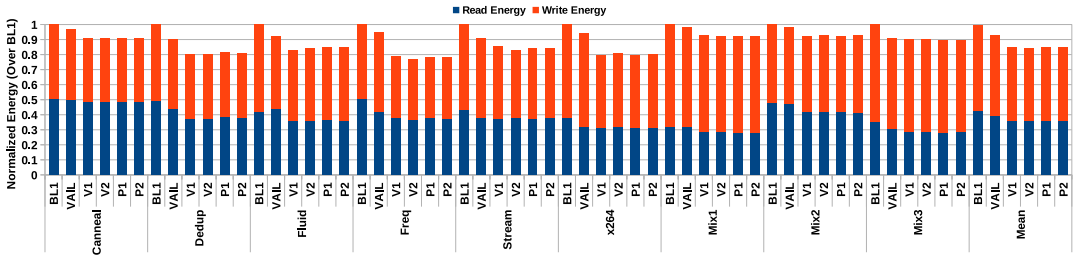


Figure 3.20: *Normalized Energy over BL1 (Lesser is better); (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

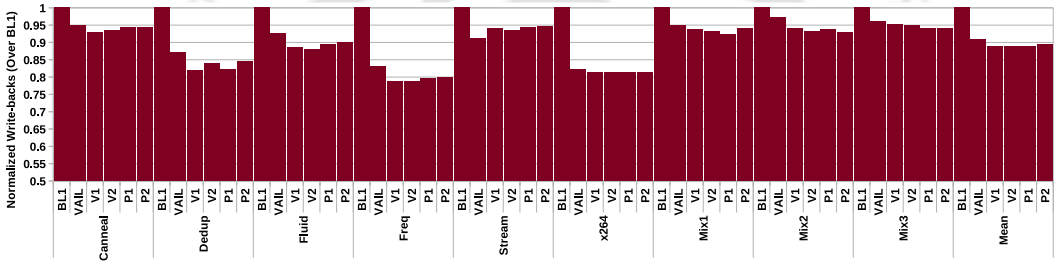


Figure 3.21: *Normalized PCM Writebacks over BL1 (Lesser is better); (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

cache and decreases write-backs to the PCM memory by retaining the blocks with more recency and more number of eviction count. However, our policies are based on exclusive hybrid memory hierarchy where DRAM and PCM are kept as part of the same layer in the overall memory hierarchy. Due to lack of this one to one correspondence of our architecture with VAIL, we have shown the comparison of the appropriate metrics separately. Figures 3.19, 3.20 and 3.21 show normalized performance (IPC), Energy and PCM writeback of BL1, our proposed techniques (VCRP, PPVC) and VAIL. Table 3.7 gives the summary of comparative analysis of our proposed techniques with VAIL.

Our techniques outperform VAIL because of the underlying difference in the idea of defining the criticality of the evicted blocks. VAIL defines a block as critical if it was evicted more recently and had a higher eviction count. However, recency and frequency of eviction give lesser insight regarding the reusability of the blocks. On the other hand, our techniques define a block as critical if it was reused enough number of times with short reuse distance during its stay at the LLC. This idea of defining criticality is completely in accordance with the temporal locality of the blocks on which the mechanism of caches is based. Our techniques make further improvisations to the critical blocks stored in the victim cache to reduce costlier PCM write-backs. VCRP works on the replacement policy of the victim cache to deliberately evict dirty DRAM blocks rather than dirty PCM blocks

Table 3.7: *Comparison of VCRP and PPVC with VAIL (All improvements are shown over BL1); (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

	% Improvement in Performance	%Reduction in Energy	%Reduction in PCM write-backs
VAIL	3.27	9.03	9.16
V1	5.65	11.78	11.28
V2	6.01	11.68	11.26
P1	5.62	11.65	11.12
P2	5.7	11.4	10.53

from the VC by scanning 2 LRU positions in search of a DRAM block. On the other hand, PPVC partitions the VC giving more share to the PCM blocks compared to the DRAM blocks to lessen PCM write-backs.

Storage and Area Overhead: In the proposed techniques, a 2-bit reuse frequency counter and 1-bit flag (per block) are used for identifying DRAM/PCM blocks of the LLC and VC. Therefore, the total storage overhead of the VC along with the counter and flag bit over BL1 (4MB cache, 4096 sets, with 64B block size and 42 bits tag) is 0.639%. Similarly, the area overhead due to association of VC with the LLC is 0.137% with respect to BL1. Note that the area overhead for the additional circuit of VC is modeled using NVSIM [129] and CACTI [130].

3.5.4 Comparative Analysis

In addition to the above results, experiments for different LLC and VC configurations were conducted.

3.5.4.1 Change in LLC Size

Table 3.8 shows the effect of the size of LLC on the performance of the existing technique WBAR and our proposed policies. It can be seen from the table that the proposed policies VCRP and PPVC perform better when the size of the LLC is small. This is because the smaller size LLCs suffer from more conflict misses than large size LLC. Therefore, storing critical blocks evicted from smaller LLCs in the VC can help in improving the performance. Note that both of our proposed policies perform better than WBAR for all sizes of LLC.

3.5.4.2 Change in DRAM:PCM Partition Ratio in PPVC

In PPVC, the VC is partitioned into smaller DRAM and larger PCM section to keep the respective blocks. This helps in retaining the critical PCM blocks, thereby reducing costly

Table 3.8: *Effect due Change in LLC Size (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

Cache Config	Policy	%Improvement in performance	%Reduction in DRAM miss rate	%Reduction in PCM miss rate	% Reduction in PCM write-backs	%Reduction in PCM reads
1MB	WBAR	2.40	1.47	7.19	6.29	13.05
	V1	7.53	19.23	19.17	14.17	18.09
	V2	7.34	19.64	19.04	14.40	18.69
	P1	7.56	18.58	18.72	14.36	18.12
	P2	7.63	18.40	18.78	13.28	18.32
2MB	WBAR	2.88	2.33	8.29	7.72	14.09
	V1	7.01	18.35	18.01	13.03	17.40
	V2	7.63	17.92	18.95	13.56	17.81
	P1	6.71	17.22	17.35	12.61	17.71
	P2	6.05	17.31	17.20	12.25	17.85
4MB	WBAR	3	2.52	9.32	8	15.1
	V1	6.65	17.54	17.75	11.28	16.02
	V2	6.01	17.5	18.19	11.27	17.09
	P1	5.62	15.11	16.49	11.13	16.5
	P2	5.7	16.52	16.28	11	17.31
8 MB	WBAR	3.5	6.33	10.82	8.5	16.38
	V1	4.8	14.9	15.38	9.75	15.4
	V2	5.03	13.2	16.12	9.52	16.6
	P1	4.32	10.02	14.03	9.88	16.23
	P2	3.8	14.02	14.42	9.48	16.41

PCM write-backs. DRAM partition helps in reducing the DRAM miss rate.

Experiments were conducted for two partition ratio (DRAM:PCM= 16:48 and 8:56) of 64 entry VC for PPVC. It can be seen from the Table 3.9 that the PCM blocks get more space in 8:56 partition as compared to 16:48 partition where the reverse is true for the DRAM blocks. As a result, the DRAM miss rate is more, and the PCM miss rate is less in 8:56 partition compared to the 16:48 partition. Performance improvement is less in the 8:56 partition than 16:48 partition. Less space for DRAM blocks in 8:56 partition results in more critical DRAM block eviction from the LLC, and VC does not have enough space to hold those blocks. It results in performance degradation in 8:56 compared to 16:48 partition. However, as the share of DRAM is only 1/4 of the total hybrid main memory (since DRAM:PCM=1GB : 3GB), the impact on the DRAM miss rate does not proportionally affect the performance. For both the scenario, there is improvement in performance over BL1 (For both variants of PPVC, on average, 5.64% and 4.60% performance improvements are achieved for 16:48 and 8:56 ratios respectively.).

Table 3.9: *Variation of DRAM:PCM Ratio in PPVC*

Ratio	Policy	%Reduction in DRAM miss rate	%Reduction in PCM miss rate	%Improvement in Performance
16 : 48	PPVC_allD	7.23	15.50	5.61
	PPVC_allD_NC	7.83	16.8	5.69
8 : 56	PPVC_allD	2.71	22.12	4.65
	PPVC_allD_NC	2.79	23.93	4.56

Table 3.10: *Effect due to Change in Number of Banks (LLC banks) in 8MB LLC (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

Number of Banks	Policy	%Imp in perf	%Reduction in DRAM miss rate	%Reduction in PCM miss rate	%Reduction in PCM reads	%Reduction in PCM wr-backs
8 Banks	BL2	2.95	8.58	7.16	4.08	4.12
	WBAR	2.17	10.97	12.27	4.49	4.84
	V1	4.81	14.54	13.38	6.64	6.17
	V2	5.72	15.18	15.88	6.61	6.77
	P1	5.20	13.86	15.24	6.23	6.09
	P2	5.39	14.34	16.27	7.21	6.35
16 Banks	BL2	2.40	8.51	8.73	5.24	3.91
	WBAR	3.38	8.33	12.47	7.09	4.03
	V1	5.60	16.82	15.50	10.84	7.30
	V2	6.54	16.35	15.23	11.43	8.39
	P1	6.12	17.38	17.58	12.18	8.01
	P2	5.65	16.71	17.09	11.55	8.08

3.5.4.3 Change in the Number of Banks in the LLC

Table 3.10 shows the effect of increasing the number of banks in the last level cache. For a particular sized LLC (8 MB in the table), as the number of banks increases (from 8 to 16), the size of each bank decreases (by a factor of 2). As discussed in the previous sections, victim caches (associated with the banks) perform better for smaller sized banks. It is reflected through lesser PCM reads and PCM write-backs of our policies in case of 16 banks compared to 8 banks. Reduction in DRAM, PCM miss rate, and eventual improvement in performance are also clear indicators of gain achieved by incorporating more number of banks in the LLC.

3.5.4.4 Change in the Number of Victim Cache entries

The size of victim cache plays a crucial role from the perspective of performance improvement, PCM writeback reduction, and overall area and storage overhead. For better system

Table 3.11: *Effect due to Change in Number of VC Entries (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

VC Entries	Policy	%Imp in Perf	%Reduction in DRAM Miss Rate	%Reduction in PCM Miss rate	%Reduction in PCM reads	%Reduction in PCM write-backs
32	V1	4.1	7.43	9.4	6.33	8.06
	V2	4.2	6.31	10.1	7.1	8.52
	P1	3.42	7.8	10	6.81	8.41
	P2	3.82	7.6	9	6.7	8.38
64	V1	5.65	17.54	17.75	16.02	11.28
	V2	6.01	17.5	18.2	17.09	11.27
	P1	5.62	15.11	16.5	16.5	11.13
	P2	5.7	16.52	16.28	17.31	11
128	V1	6.71	18.06	17.81	17.01	11.9
	V2	6.5	18.24	18.47	17.1	12.02
	P1	6.2	16.9	16.73	17.05	12.04
	P2	6.1	19.85	16.41	17.5	11.21

performance and costly PCM writeback reduction, it is always tempting to increase the size of the victim cache. However, as discussed in the sensitivity analysis, increasing victim cache size does not return rewarding benefits after a certain size of victim cache. We term this phenomenon as the *Law of Diminishing Returns of Victim Cache.* Table 3.11 shows the effect of changing the number of VC entries (for 32, 64 and 128 entries) for different metrics. It is evident from the table that benefits obtained from victim cache associated with LLC saturate when its size goes beyond 64 entries. Since large size of VC also aids to the large area and storage overhead; therefore the optimal cut-off size of victim cache is decided to be 64 entries.

3.5.4.5 Change in DRAM : PCM Ratio in Hybrid Main Memory

Exclusive hybrid memories comprise of DRAM and PCM as parts of the same main memory layer. Generally, the capacity of the DRAM portion is much lesser than the PCM portion in order to get the latency benefit of DRAM and the high capacity benefits of the PCM region. We have taken DRAM: PCM ratio as 1GB : 3GB (giving lesser space to DRAM and more space to PCM) to realize a highly dense hybrid memory system that does not have to sacrifice the latency benefits of DRAM. However, in order to give an idea of how changing DRAM : PCM ratio would affect our proposed techniques, experiments were conducted for hybrid memories with DRAM : PCM ratio 2GB : 2GB. Table 3.12 presents a comparison of performance improvement and energy reduction (over BL1) of our proposed techniques

Table 3.12: *Effect on Changing DRAM:PCM Ratio in Hybrid Main Memory (V1 : VCRP_**allD**, V2 : VCRP_**allD_NC**, P1 : PPVC_**allD**, P2 : PPVC_**allD_NC**)*

DRAM:PCM Ratio	Techniques	%Imp in perf	%Red in Energy
1 : 3	V1	5.65	11.78
	V2	6.01	11.68
	P1	5.62	11.65
	P2	5.7	11.4
2 : 2	V1	10.51	7.15
	V2	10.7	7.40
	P1	10.2	7.45
	P2	10.31	7.25

for DRAM : PCM ratio 1 : 3 and 2 : 2, respectively.

It is straightforward to follow from the Table 3.12 that improvements in performance is more when the DRAM : PCM ratio is 2 : 2 than 1 : 3. Improvement in performance is due to increase in the DRAM portion in 2 : 2 hybrid memory (increase from 1 GB to 2 GB). The DRAM read/write latencies are less compared to PCM read/write latencies, which help in improving performance when the share of DRAM partition is more in the hybrid memories. On the other hand, PCMs reads/writes are costlier from the perspectives of energy consumption. Therefore, our techniques that aim to reduce both PCM read and writes offer more rewarding results in terms of reduction in energy consumption when the share of PCM is more in the hybrid memory. It is reflected in more reduction in energy (over BL1) when the DRAM : PCM ratio is 1 : 3 as compared to 2 : 2. Considering today's demand for more scalable and highly dense memory systems, it is preferable to adopt hybrid memory with 1 : 3 ratio as it is highly dense compared to 2 : 2 hybrid memory. However, results confirm that our proposed techniques bring impressive improvements in performance and energy consumption for both types of hybrid memories (DRAM : PCM=1 : 3 and 2 : 2).

3.6 Summary

Hybrid main memory composed of DRAM and PCM show immense potential for being replacement choices of DRAM only main memory. LLC management policies act a major role for efficient utilisation of these hybrid memories. However, the traditional LLC management policies whose only yardstick for a better system performance is improving the hit rate, are oblivious of the underlying disparity in the miss cost of DRAM and PCM blocks. Therefore, these policies cannot maintain similar performance as DRAM-only memories for

the hybrid main memories.

In this chapter, two techniques are proposed based on the use of a small victim cache associated with the LLC. Victim cache being a limited resource, for both the techniques, only critical blocks are put in the VC based on the concept of reuse distance. The criticality of the blocks is defined using the concept of reuse distance and frequency of the block accesses. The first technique VCRP works on the replacement policy of the VC. In contrast, the second technique PPVC gives a larger share of VC to the PCM blocks over the DRAM blocks by logically partitioning the VC. Our techniques: ⟨VCRP, PPVC⟩ reduce PCM-writes by ⟨11.27%, 10.82%⟩ and ⟨4.86%, 4.4%⟩ over BL1 and WBAR respectively. In doing so, our techniques do not have to sacrifice DRAM miss rate and obtain ⟨17.52%, 15.81% ⟩, ⟨85.6%, 84%⟩ improvement over BL1 and WBAR respectively; and PCM reads are reduced by ⟨16.25%, 16.1%⟩, ⟨8.73%, 8.56%⟩ over BL1 and WBAR respectively that otherwise might have negative impact on the overall system performance. Coverage of all aspects of improvement exploiting every nook and corner leads to performance improvement of ⟨5.83%, 5.66%⟩ and ⟨3.35%, 3.17 %⟩ and energy reduction of ⟨11.73%,11.52%⟩ and ⟨4.75%, 4.53%⟩ for ⟨VCRP,PPVC⟩ over BL1 and WBAR respectively. Thus, designing LLC management techniques taking into account the disparity of the latencies of hybrid main memories can help in smoother integration of newer technologies in the memory hierarchy.

4

SWEL-COFAE : Write Reduction Using Compression and Adaptive Encoding Augmented by Wear Leveling

In this chapter, we propose a word-level compression scheme called COMF to reduce bit-flips in PCMs by removing the most repeated words from the cache blocks before writing into memory. COMF is augmented with an adaptive granularity-based encoding technique to form COFAE, which reduces bit-flips to a further extent. We also propose SWEL-COFAE, an intra-line stride-based wear leveling technique to improve PCM lifetime by balancing the bit-flip pressure within the cells of the memory lines. Experimental results show that the proposed technique improves lifetime by 101% and reduces bit-flips and energy by 59% and 61%, respectively over baseline.

4.1 Introduction

As discussed earlier, bit-flips occur in the PCM cells as a result of writing the blocks that are evicted from LLC. It degrades memory lifespan and increases energy consumption. Therefore, to increase durability and reduce energy consumption in PCM, it is required to minimize write activity at the bit level (bit-flips).

Several techniques have already been proposed to improve the lifetime of the NVM-based main memories. These techniques can be broadly classified into two classes: 1) *Write*

Activity Reduction techniques that reduce bit-flips in NVM main memories and 2) *Wear Leveling* techniques that distribute the writes evenly across the memory space to minimize the chances of failure of the individual cells due to excessive write pressure. Techniques such as *Data Comparison Write* (DCW) [42] and *Flip-N-Write* (FNW) [43] are encoding techniques that focus on reducing bit-flips in PCM-based main memories. In DCW, only the modified bits of the cache blocks are written into the memory to reduce unnecessary bit-writes. FNW is an enhancement of DCW, which bounds the maximum bit-flips to half of the number of bits in the blocks by inverting the data if more than half of the bits in the blocks are modified.

We proposed a compression technique called COMF that removes the repeated words from the incoming cache blocks before writing to the PCM-based main memory. It reduces bit-flips and improves PCM lifetime as the amount of data written to the PCM cells becomes lesser. However, COMF can be easily augmented with encoding techniques that promote further reduction in bit-flips. Therefore, we propose an adaptive encoding-based technique that encodes the compressed blocks generated by COMF to further reduce bit-flips. It helps in boosting the PCM lifetime due to reduced write activities in the PCM cells. In order to improve the PCM lifetime further, two intra-line wear leveling techniques are also proposed that balance the bit-flip pressure by periodically changing the orientation and writing positions of the compressed data in the PCM memory lines.

4.2 Chapter Overview

4.2.1 Contributions of the Chapter

The main contributions of the chapter are as follows:

- We describe in detail our proposed Word-level compression technique COMF based on the removal of most frequently occurring words from the cache blocks.
- We propose an adaptive encoding technique on the compressed data produced by COMF. It encodes the bits in the compressed blocks opportunistically at a finer granularity. It helps in reducing the bit-flips further. The combined technique of COMF and Adaptive Encoding is termed as COFAE.
- We propose two wear leveling techniques in incremental fashion to balance the bit-flip distribution within the memory lines. The first technique, called *Orientation-based* wear leveling changes the orientation of writing the compressed block periodically. In contrast, the second technique called *Stride-based* wear leveling changes the writing position of the compressed blocks periodically to shift the bit-flip pressure towards the less write activity-prone middle cells of the memory lines. This integrated solution which combines Stride-based Wear Leveling with COFAE is termed as SWEL-COFAE.

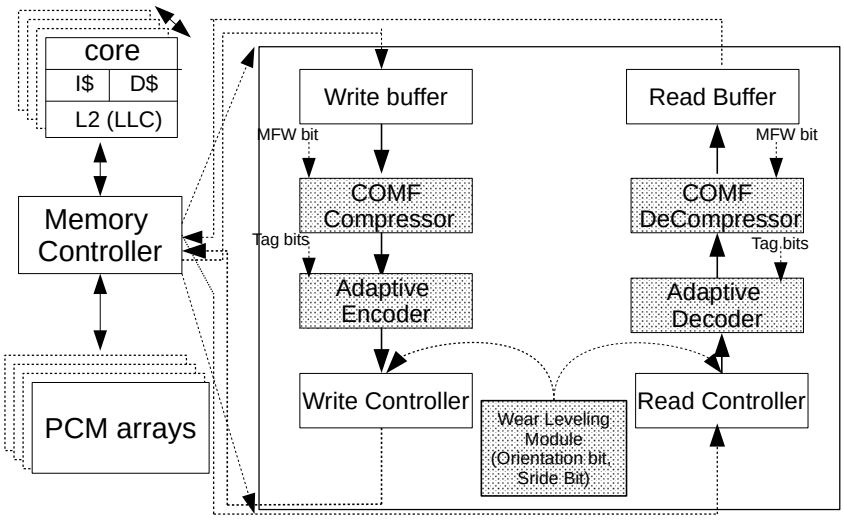


Figure 4.1: *Architecture of the Proposed Scheme : SWEL-COFAE (Shaded Part indicates our contribution)*

- A detailed analysis of hardware overhead in terms of latency, area, and power of the COMF compressor and decompressor is also given.

4.2.2 Chapter Organization

The rest of the chapter is organized as follows. Proposed methodology is discussed in Section 4.3. Section 4.4 illustrates the experimental evaluation, followed by summary in Section 4.5.

4.3 Proposed Methodology

Our proposed methodology is based on the integration of three different techniques related to compression, data encoding, and wear leveling. The high-level view of the architecture of the proposed methodology is shown in Figure 4.1. Our proposed compression algorithm reduces the amount of the data written in PCM by compressing the blocks incoming to the PCM main memory. Association of Adaptive Encoding on these compressed blocks further reduces bit-flips in the PCM cells. Finally, the wear leveling [1] algorithm acts on

[1] In this work, we focus on intra-line wear leveling to improve the lifetime PCM.

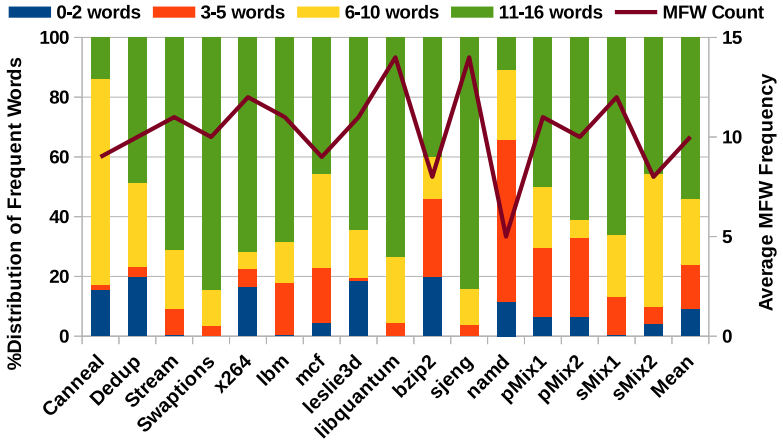


Figure 4.2: *Percentage Distribution of Repeated Words*

the blocks (generated after Compression and Adaptive Encoding) to balance the bit-flip pressure within the PCM cells.

4.3.1 COMF : Word-Level Compression Scheme

The cache blocks evicted from the LLC and incoming to the main memory have many repeated words. These blocks can be compressed by removing the most frequently occurring words before writing into the PCM main memory. In order to measure the percentage of repeated words in the cache blocks, we conducted an experiment for the various PARSEC [116], and SPEC 2006 [133] benchmarks. Figure 4.2 shows the percentage break up in the number of repeated words for these benchmarks. As it is evident from the figure, most benchmarks have a high percentage of repeated words in their cache blocks. The percentage of words occurring between 11 to 16 times in a cache block (containing 16 words) is around 54%. Similarly, the percentage of words repeating 6-10, 3-5, 0-2 times in a cache block are 22%, 15%, and 9%, respectively. The line graph in Figure 4.2 shows the average frequency of the Most Frequently occurring Word (MFW) in the cache blocks for different benchmarks, which is also very high (on average=10). These highly repeated MFWs can be removed from the cache blocks (by keeping only the first word among those and removing the later occurrences) to compress the data.

We propose a word-level compression technique called COMF for reducing bit-flips in the PCM-based main memory. Algorithm 3 shows the compression and decompression algorithms of COMF. As shown in Algorithm 3, a compression bit ($MFWbit$) is used that tells whether the cache block is compressible or not. A 4-bit field $MFWindex$ is used to store the index of the first occurrence of the MFW. Finally, a field called $MFWtag$, consisting of 16 bits (1 bit/word), is used to identify whether a particular word is included

Algorithm 3: COMF Compression-Decompression

```
1  n_uc : Total words in the uncompressed cache line
2  b_uc: Uncompressed cache line data, b_c : Compressed cache line data
3  MFWbit : MFW Compression bit, MFWindex : MFW index in b_uc
4  MFWtag : MFW tag array
5  w_i : ith word inside a cache line, w_MFW : Most frequent word
6  freq(w_i): Frequency of the ith word, maxfreq : Maximum frequency of a word in a cache line
7  Function Compression(b_uc)
8      for w_i ϵ b_uc , i ϵ 1,2,....n_uc do
9          Calculate freq(w_i)
10         Find maxfreq, MFWindex
11     if maxfreq > th then
12         b_c.MFWbit = 1
13         b_c.MFWindex = MFWindex
14         for ∀w_i ϵ b_uc do
15             if ((freq(w_i)! = maxfreq)||(freq(w_i)==
16             maxfreq & i == MFWindex)) then
17                 b_c.MFWtag[i] = 1
18                 Add w_i to b_c
19             else
20                 b_c.MFWtag[i] = 0
21     return b_c
22 Function DeCompression(b_c)
23     for i = 1 to n_uc do
24         if b_c.MFWtag[i] == 1 then
25             Add next word of b_c to b_uc
26         else
27             Add w_MFW to b_uc
28     return b_uc
```

in the compressed form or not. The $MFWindex$, and $MFWtag$ together constitute the meta-data inside the compressed block. For every incoming block, the frequency of the words in the block is calculated (shown in line 9). Then, the maximum frequency of the words ($maxfreq$) and the index of the MFW ($MFWindex$) are determined (line 10). We allow the compression of the cache blocks in which the maximum frequency of occurrence of a word is more than a threshold [1] th (line 11). For such cache blocks, we set the compression bit $MFWBit = 1$ (line 12) and set the $MFWindex$ with the index of the first occurrence of the MFW (line 13). Then, with a single scan of all the words in the uncompressed block, all the repeated MFWs are removed except its first occurrence. $MFWtag$ for a particular word position is set as 1 if that word is actually included in the compressed block (lines 17, 18). Otherwise, the $MFWtag$ for a word position is set to 0 if that word is removed in the compressed block (line 20).

During decompression, the outgoing blocks are decompressed to bring them to their original uncompressed form. The decompression is done by scanning the $MFWtag$ array. If $MFWtag[i] = 1$ for a particular word index i, then the next word from the compressed block is added to the decompressed block (lines 24-25), else if $MFWtag[i] = 0$, the most frequent word is added as the next word (lines 26-27). Then, the decompressed block is returned to the requester LLC (line 28). COMF is a lossless compression technique as the compressed data can be decompressed to the exact original form using the associated meta-data.

Working Example

Figure 4.3 shows a working example of COMF based compression and decompression scheme. The uncompressed cache block is assumed to contain 8 words (each of 4 bytes) for illustration purposes. The word `0X00007610` is the MFW with a frequency of 5, and it occurs for the first time at index 2. Since MFW frequency is more than th (= 3, taken for the example), the cache block is compressed by removing the repeated words. Accordingly, $MFWbit$ is set to 1, and $MFWindex$ is set to 2. Finally, the $MFWtag$ consisting of 8 bits are set as `11110000` since the most frequent word `0X00007610` from $4th, 5th, 6th$ and $7th$ indices are removed from the compressed block while keeping only its first occurrence at index 2. Therefore, the original cache block consisting of a total of 256 bits reduces to 144 bits (=Metadata(16 bits)+Compression Data area (128 bits)) after compression. For decompression, the algorithm is executed in the reverse direction by constructing the block using the meta-data and the compressed data.

[1] We did extensive empirical analysis for different values of th and found that $th = 8$ is the optimal value (cf. Section 4.4.2.1).

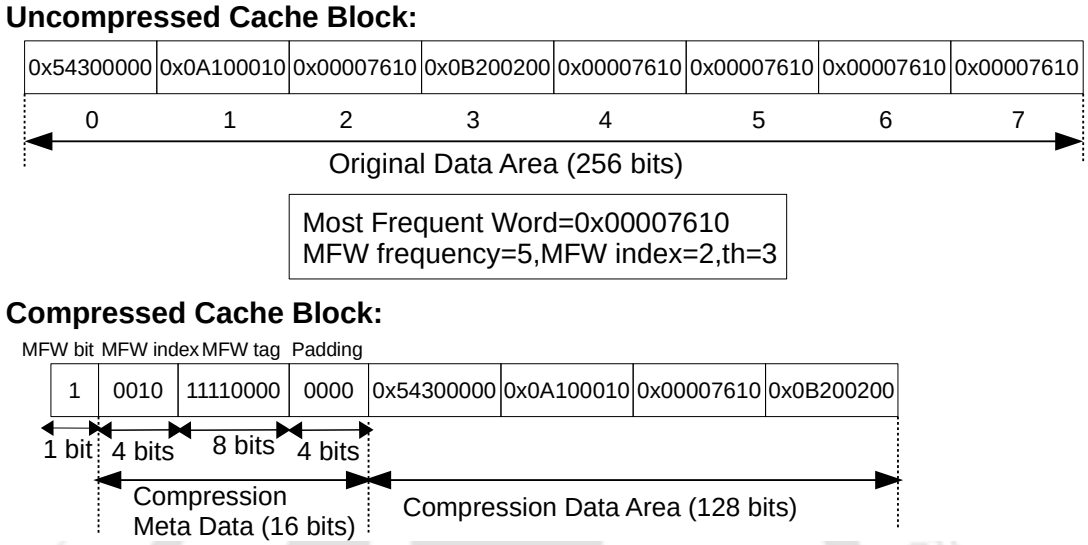


Figure 4.3: *Working Example of COMF*

4.3.2 COFAE : Proposed Adaptive Encoding on the COMF Compressed Blocks

In this section, we describe our new Adaptive Encoding based technique applied on the compressed blocks generated by COMF. The integrated solution of COMF and Adaptive Encoding is termed as COFAE. We present some preliminary concepts first in Sections 4.3.2.1 and 4.3.2.2 before explaining the core idea of COFAE.

4.3.2.1 Encoding Techniques

Various data encoding techniques [43, 134] have been proposed to mitigate the issues of low write endurance and high write energy in PCM main memories. These techniques reduce the bit-flips in PCM by transforming the bit-streams to be written in the PCM cells. We explain the working of one of the most popular encoding techniques, FNW [43] with an example.

In FNW, each data block is divided into partitions, and tag bits are assigned to govern the bit-flips within each partition. While writing new data, if the number of bit-flips in a partition is more than half of the partition size, the new data bits are written in the inverted form, and the corresponding tag bit is set to 1. Otherwise, the bits in the partition are written as it is, and the tag bit is reset to 0. The number of data bits represented by a tag bit (i.e., number of bits within a partition) is termed as *Encoding Granularity*. The formula for obtaining *Encoding Granularity* ($Gran_{Fixed}$) of FNW that assumes a fixed granularity is given by the Equation 4.1, where, b_{uc} is the number of bits in an uncompressed cache block,

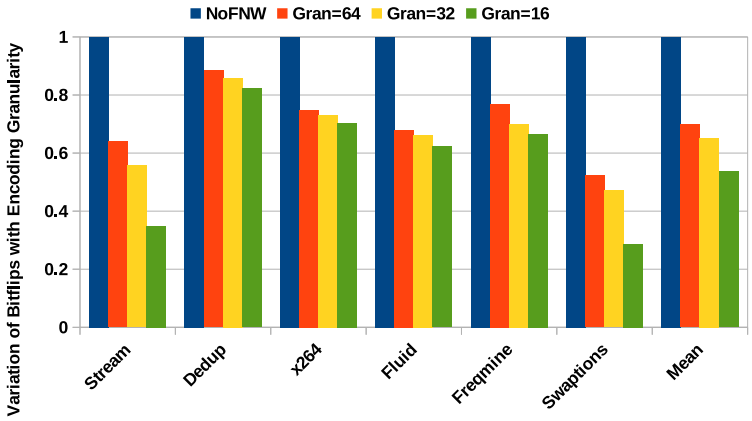


Figure 4.4: *Bit-flips Reduction for Varying Granularity in Non-Adaptive Encoding*

and T is the number of tag bits assigned for the block.

$$Gran_{Fixed} = \frac{b_{uc}}{T} \tag{4.1}$$

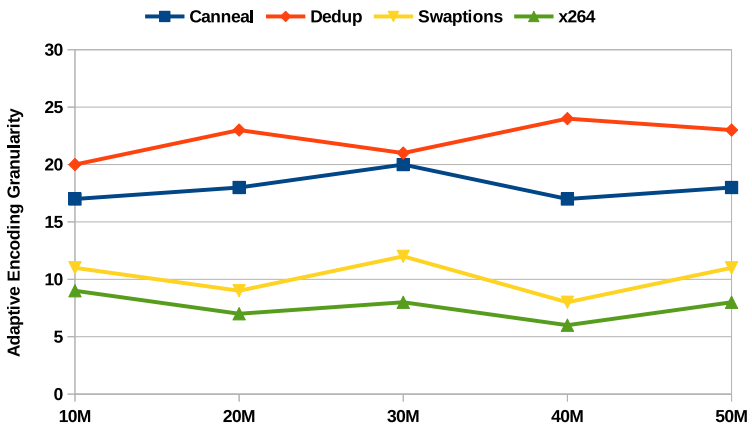


Figure 4.5: *Variation of Adaptive Granularity over Time (calculated using Equation 4.2)*

4.3.2.2 Adaptive Encoding Granularity

Studies [43, 44, 134] show that as the *Encoding Granularity* becomes finer (smaller), the reduction in bit-flips by the encoding techniques increases. We conducted experiments for varying encoding granularities ($Gran_{Fixed}$ = 64, 32, and 16) to compare the reduction in bit-flips achieved with respect to a baseline where no encoding technique is applied. Figure 4.4

shows that the reduction in bit-flips becomes more when the encoding granularity becomes finer. On average, reduction in bit-flips are 30%, 36%, and 46% for $Gran_{Fixed}$=64, 32, and 16, respectively, over the baseline where no encoding technique is applied. However, more tag bits are needed to make *Encoding Granularity* finer, which increases the storage overhead.

The traditional FNW encoding that uses a fixed encoding granularity can be made to perform more optimally if it is preceded by a compression technique. Compression results in reduced size of the cache blocks; therefore, the tag bits for encoding could be utilized more efficiently by assigning them to the compressed data bits only. As a result, the number of data bits represented by a tag bit reduces due to compression, resulting in a finer *Encoding Granularity* with the same number of tag bits depending on the size of the compressed block (more the compression is, more finer is the Encoding Granularity). We define such encoding granularity obtained by adaptively assigning the tag bits to the compressed data as the *Adaptive Encoding granularity*. Note that the fixed granularity-based Non-adaptive encoding has to increase the number of tag bits to achieve finer encoding granularity, which increases the storage overhead. Figure 4.5 shows the variation of *Adaptive Encoding Granularity* over time when FNW encoding is preceded by COMF, where $Gran_{Fixed} = 32$. Observing the trends for all the benchmarks, we can say that the *Adaptive Encoding Granularity* is fairly below $Gran_{Fixed}$ (i.e., finer than $Gran_{Fixed}$) during the entire time frame of 50M cycles. It indicates a clear edge of adopting Adaptive Encoding over fixed granularity-based Non-Adaptive Encoding. Therefore, we can achieve substantially finer encoding granularity (hence more reduction in bit-flips) by applying adaptive encoding over the compressed blocks generated by COMF using fewer tag bits compared to Non-adaptive encoding. The formula for calculating *Adaptive Encoding Granularity* ($Gran_{AE}$) is given by equation 4.2, where b_c is the number of bits after the block is compressed.

$$Gran_{AE} = \frac{b_c}{T} \tag{4.2}$$

4.3.2.3 COFAE

Our proposal COFAE applies an Adaptive Encoding technique on the compressed cache blocks produced by COMF. It reduces bit-flips by achieving finer granularity compared to the traditional Non-Adaptive Encoding technique. In particular, once the block is compressed using COMF, we apply $Gran_{AE}$ (Equation 4.2) to decide the way the block gets written to the PCM memory.

Due to compression, some portion of the block does not get written, and hence the tag bits need not govern them. The bit-flips will reduce as the tag will point to a smaller portion of the block. Note that the number of tag bits is fixed, but the granularity of data that

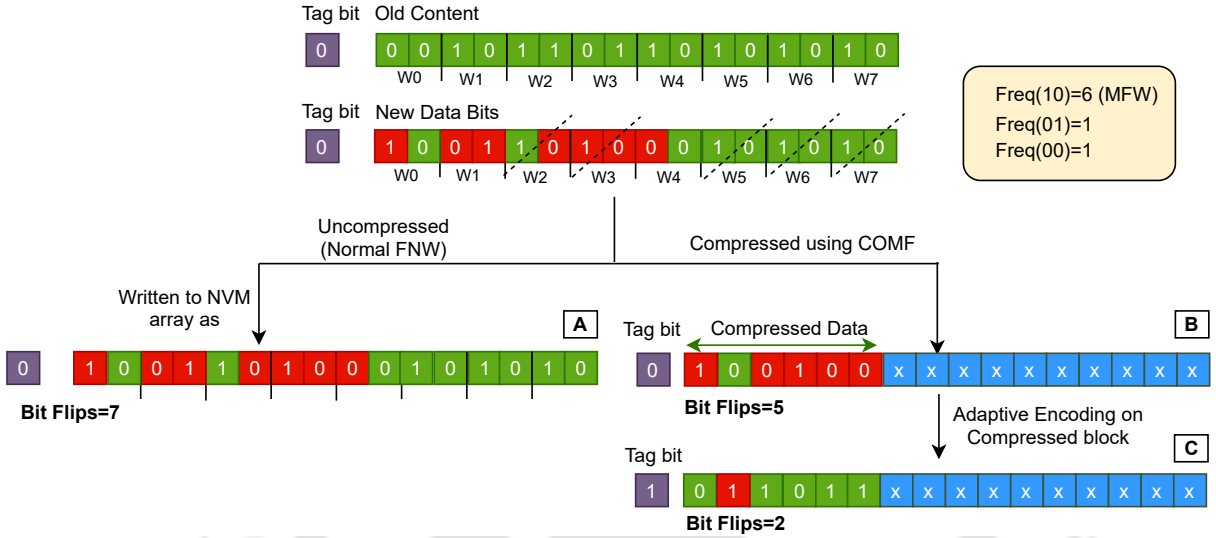


Figure 4.6: *Working Example of Adaptive Encoding on COMF Compressed Blocks (Bits shown in red boxes show the flipped bits with respect to the old data content)*

they represent changes at runtime depending on the size of compressed data. The size of the compressed block can be easily derived using the meta-data of COMF.

Working Example of COFAE

Figure 4.6 shows a comparison between FNW-based Non-Adaptive Encoding and Adaptive Encoding technique-COFAE. The length of data bits is taken to be 16 bits for illustration purposes. The bit-stream consists of 8 words (w0, w1,...w7), each of 2-bits, as shown in the figure. One tag bit is assigned that governs all the 16 bits, accounting for $Gran_{Fixed}$=16. The right side of the figure shows the effect of applying adaptive encoding on the compressed data produced by COMF. The left side of the figure shows the effect of Non-adaptive encoding when applied on the bit-stream without compression.

During writing the new data content, if FNW is applied on an uncompressed block (shown on the left side of the figure), the number of data bit-flips is 7 (shown in red boxes), which is less than $Gran_{Fixed}/2$. Therefore, the new bits are written as it is, and the tag bit is reset to 0 (Block labeled with Box A). The total bit-flips is 7 (Tag bit-flips+Data bit-flips=0+7=7).

When COMF is applied to the new bit-stream, it compresses it by removing the most frequent words, which happens to be words w2, w3, w5, w6, and w7 with content <10> and frequency 6 (we assume compression threshold, th=3 for this example; MFW frequency(6)>th), while keeping only its first occurrence at index 0 (w0). COMF reduces the bit-stream to 6 bits (<100100>) from an initial length of 16 bits after removal of the MFWs. Since the data bits of only the compressed block are written in the memory, therefore, the other bits need

not be written in the PCM array. As a result, the tag bit can be assigned adaptively to the compressed area (first 6 bits) only, resulting in $Gran_{AE}$=6. The number of flipped bits in the compressed area (=5) is more than $Gran_{AE}/2$. Therefore, the compressed data bits are written in the inverted form, and the tag bit is set to 1 (Box C in Figure). It results in a total 2 bit-flips (Tag bit-flip+ Data bit-flips=1+1=2).

Suppose we use fixed granularity ($Gran_{Fixed}$) on the compressed data, the bit-flips (=5) are less than $Gran_{Fixed}/2$; it will result in a total of 5 bit-flips (Box B).

It demonstrates that our adaptive encoding approach- COFAE can reduce bit-flips to a higher degree than the traditional Non-Adaptive encoding technique that adheres to a fixed encoding granularity. Note that COFAE does not need to increase the number of tag bits to achieve finer encoding granularity, unlike Non-Adaptive Encoding.

4.3.3 SWEL-COFAE : Proposed Intra-line Wear Leveling Techniques

COMF reduces the size of the cache blocks to be written to a great extent. However, the compressed portion occupies the upper half of the entire cache block, and the saved space due to compression occupies the lower half. It creates an uneven write pressure within the PCM cells over several writes. In order to uniformly distribute the bit-flips within the memory lines, we propose two wear leveling techniques. Below, we discuss the techniques in detail.

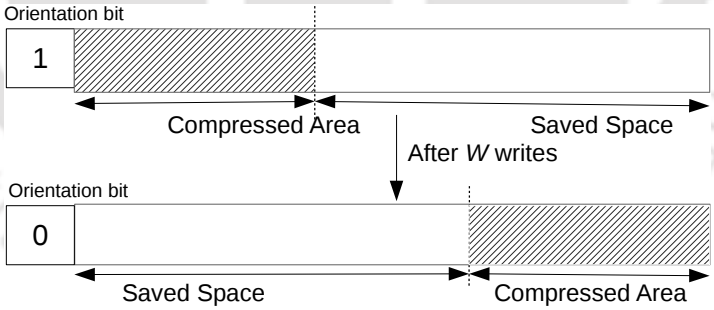


Figure 4.7: *Working Example of Orientation-based Intra-line Wear Leveling*

4.3.3.1 Orientation-based Wear Leveling

It periodically changes the orientation of writing the compressed lines in the PCM arrays after a specific number of writes to the line. Figure 4.7 shows a pictorial view of the functioning of the wear leveling technique. We keep a bit called, *Orientation* bit that tells

Table 4.1: *Average Bit-flip Distribution at Different Bit Positions in a Memory Line by Changing Orientation Periodically*

	0 -99	100 -199	200 -299	300 -399	400 -512	Std. Dev (σ)
Dedup	0.26	0.17	0.17	0.16	0.24	0.044
x264	0.23	0.17	0.17	0.18	0.25	0.039
Fluid	0.25	0.21	0.12	0.15	0.27	0.064
Canneal	0.29	0.18	0.13	0.16	0.24	0.062
Stream	0.22	0.16	0.15	0.20	0.27	0.047
Mean	**0.25**	**0.17**	**0.14**	**0.16**	**0.26**	**0.048**

the order of writing. If the *Orientation* bit is 1, the compressed line is written in the *left-to-right* order, thus compressed portion occupies the upper half portion. However, after a specific number of writes (W), the *Orientation* is set to 0, and then the next writing takes place in the *right-to-left* order. This periodic change in the order of writing takes place after every W number of writes.

However, with this orientation-based wear leveling approach, the cells corresponding to the middle bits inside a memory line [1] experience relatively lesser bit-flips than the cells towards the two extremes of the memory line. Table 4.1 shows the distribution of bit-flips across different zones of bit locations inside the memory lines (512-bit width) for some representative workloads. Average normalized bit-flips in the bit positions 0-99, 100-199, 200-299, 300-399 and 400-512 bit are 0.25, 0.17, 0.14, 0.16 and 0.26 respectively, accounting for an average standard deviation (σ_{avg}) of bit-flips across these zones equal to 0.048. It clearly shows that the cells corresponding to the bit positions (0-99 and 400-512) towards the two ends of a memory line experience many more bit-flips compared to the middle cells [2].

4.3.3.2 Stride-based wear leveling

The uneven writes within the memory line can be balanced further by shifting the position from where the compressed block gets written. We call this shifted distance as the *Stride Distance*, and the enhanced wear leveling technique as Stride-based wear leveling. The integrated solution of COFAE and **S**tride based **WE**ar **L**eveling is termed as SWEL-COFAE. In this technique, a bit called *Stride Bit* is kept per cache block that tells whether the block is to be written from the ends (when *Stride Bit*=0) or from the *Stride Distance* (when *Stride Bit*=1).

[1] A memory line refers to an array of PCM cells where a cache block gets fitted.

[2] Since the compression threshold for COMF is 8 (half of the total words in a cache block, Discussed in Section 4.4.2.1), COMF produces highly compressed blocks

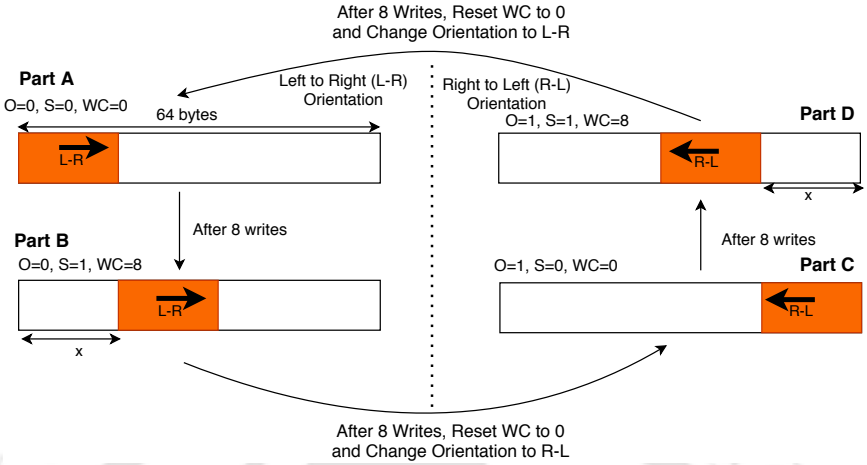


Figure 4.8: *Working Example of Stride-based Wear Leveling, O: Orientation Bit, S: Stride Bit, WC : Write Count, x: Stride Distance*

Figure 4.8 shows the working of Stride-based wear leveling. While writing a compressed block in the memory line, the Orientation Bit (O) and the Stride Bit (S) are checked. Initially, these bits are reset to zero for all blocks. Also, the write counter (WC) associated with the block is reset to 0. The actions taken for different combinations of O and S are discussed below:

i) O=0, S=0: Initially, the O and S bits for a block are 0. The orientation of writing is taken to be *left-to-right* (L-R) (since O=0), and the block is written from the left end of the memory line (since S=0) (As shown in Part A of the Figure 4.8).

ii) O=0, S=1: S is set to 1 after 8 writes to the memory line. From the next write onwards, the compressed block is written with the *left-to-right* orientation from the *Stride Distance* (shown in Part B, Figure 4.8).

iii) O=1, S=0: After another 8 writes (a total of 16 writes) to the memory line, the orientation of writing is changed to *right-to-left*. The bit O is set to 1, write count (WC) is reset to 0, and S is reset to 0. The block is written with a *right-to-left* orientation starting from the right end of the memory line (shown in Part C, Figure 4.8).

iv) O=1, S=1: After 8 writes, the S is set to 1. From the next write onwards, the block is written in *right-to-left* order starting from the *Stride Distance* (Shown in Part D, Figure 4.8). Finally, after 8 writes (i.e., a total of 16 writes to the memory line in *right-to-left* orientation), O, S, and WC are reset to 0, and the process is repeated. Therefore, the write pressure is distributed across all the cells of the memory line.

During a read operation, the (O, S) bit pair for the block determines the retrieval of the correct bytes of the block from memory. Finally, the original uncompressed block is returned after decoding and decompressing. Flowchart in the figure 4.9 shows the operations done during read and write path by SWEL-COFAE.

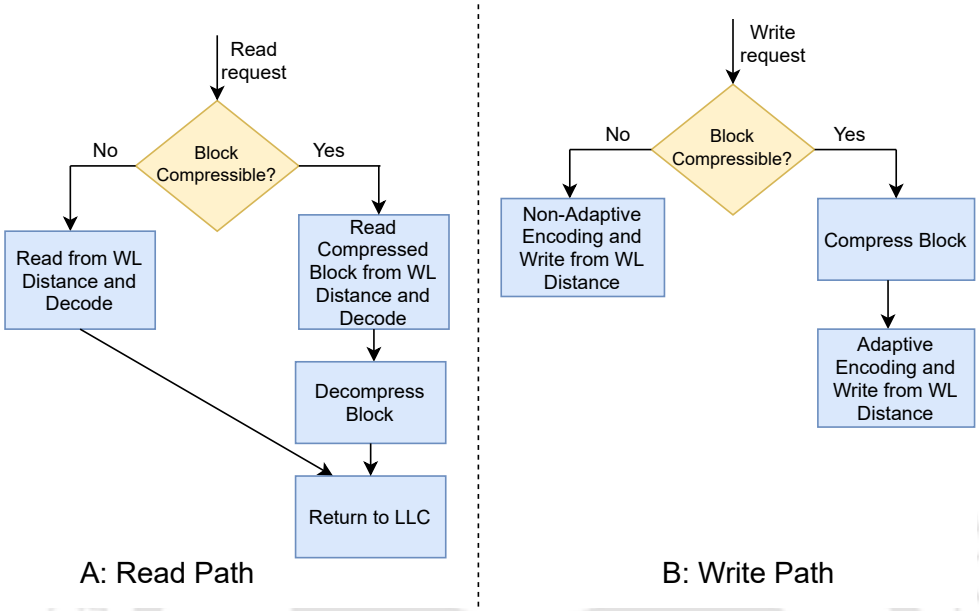


Figure 4.9: *Flowchart Showing the Read Path and Write Path for SWEL-COFAE; WL : Wear Leveling*

Table 4.2: *System Parameters and Benchmarks*

Components	**Parameters**
Processor	2Ghz, Quad-core, X86
L1 Cache	Private, 32 KB SRAM split I/D caches, 2-way associative, 64 B block, 1 cycle latency
L2 Cache (LLC)	Shared, SRAM, 64B block, 8-way associative 10 cycle latency, Size: 8MB
Main Memory	PCM: 4 GB, 4 channels, Memory Controller: FRFCFS, Page size : 4KB
Memory latency [50]	PCM :: Read latency: 50ns, Write latency: 150ns
(De)Compression latency	Compression Latency : 3.5 ns, Decompression Latency : 2.65 ns
Memory Energy [21]	PCM :: Read Energy (pJ/bit): 2.47 Write Energy (pJ/bit): 14.03 (SET), 19.73 (RESET)
Benchmarks: PARSEC: Canneal, Dedup, x264, Swaptions, Streamcluster SPEC 2006 : lbm, mcf, leslied3d, libquantum, sjeng, bzip2, namd Mixes: **pMix 1:** Canneal, Dedup, Freq, Stream ; **pMix 2:** Freq, Fluid, Stream, x264; **sMix 1**: gobmk, lbm, sjeng, namd ; **sMix 2**: milc, mcf, hmmer, bzip2	

Table 4.3: *Analysis of the benchmarks in terms of Memory Intensity and Word Locality : HWL/LWL/MWL = High/Low/Medium Word Locality*

Benchmark	MPKI	WBPKI	Word Locality (%)	Type
Canneal	3.4	1.9	56.25	LWL
Dedup	0.34	0.23	62.5	LWL
Stream	0.8	0.7	75	HWL
Swaptions	0.02	0.03	81.25	HWL
x264	2.6	0.9	75	HWL
lbm	25	18	68.75	MWL
mcf	23	6	56.25	LWL
leslie3d	9	6	68.75	MWL
libquantum	6.9	6.8	75	HWL
sjeng	8.5	8.3	81.25	HWL
namd	0.09	0.02	43.75	LWL
bzip2	8.8	5.7	56.25	LWL
gobmk	27.82	20.59	75	HWL
milc	5.68	1.62	68.75	MWL
hmmer	0.3	0.04	37.5	LWL

4.4 Experimental Evaluation

We implemented our technique on a full system simulator GEM5[124] integrated with NVMain[125], a cycle-accurate main memory simulator designed for NVMs. The system parameters used in the experiments are shown in Table 4.2. We evaluated our results using multi-threaded PARSEC [116] and multi-programmed SPEC 2006 [133] benchmark suite, as shown in Table 4.2. We selected the workloads based on memory intensity (read/write intensity) and word locality, as shown in Table 4.3. The read and write intensity are measured in terms of Misses Per Kilo Instruction (MPKI) and Write-Back per Kilo Instruction (WBPKI), respectively. On the other hand, word-locality is defined as the percentage frequency of the MFW in the incoming blocks. Therefore, high-word locality of a benchmark essentially leads to the generation of highly compressed blocks by COMF. We label the benchmarks as High-Word-Local (HWL)/Medium-Word-Local(MWL)/Low-Word-Local [1] depending on the word locality. The mix benchmarks are composed by considering the Write-Backs per Kilo Instruction (WBKI) of each individual benchmark. We run each SPEC workload for 1 Billion instructions by warming them up by 250M instructions. We have taken *Stride Distance*=16, i.e., 1/4th of the size of a cache block (64 bytes) for SWEL-COFAE, keeping in view the highly compressed blocks generated by COMF, since it can offer maximum

[1]Word-locality range of HWL, MWL, and LWL are => 75%, 65 – 75% and <= 65%, respectively.

balancing of bit-flips across the PCM memory cells.

We evaluate the following schemes for analysis. Out of these, six are existing techniques: DCW, FPC, E1, E2, E3, and READ. Note that all the schemes use DCW [42] to reduce redundant bit-flips.

- **DCW [42]:** Only the modified bits of the cache blocks are written to PCM main memories to eliminate redundant bit-flips.
- **FPC [89]:** It compresses the cache blocks based on patterns stored in a pattern table. Bit-flips are reduced as the amount of data written to PCM is reduced due to compression.
- **E1 (FPC with Wear Leveling) [49]:** It compresses the cache blocks by FPC[89] and then introduces an intra-word wear leveling technique to increase the lifetime of PCM further.
- **E2 (FPC with Encoding) [50]:** It compresses the cache blocks using FPC [89]. Then, it opportunistically combines encoding techniques FNW [43] or FlipMin [134], depending on the saved space, to reduce the bit-flips.
- **E3 (FPC/BDI with Encoding) [51]:** For an incoming block, the sizes of the compressed blocks generated by FPC and BDI compression are determined. Then, the block is compressed using the compression technique that produces smaller size block. Finally, encoding is applied on the compressed data.
- **READ [44]:** It reduces bit-flips in PCM-based main memories by using fine granularity encoding, obtained by eliminating the writes of the clean words from the cache blocks.
- **HWL from DEUCE [39]:** Performs intra-line (Horizontal) wear leveling using algebraic start-gap method.
- **COMF :** Proposed method to compress block depending on the most frequently occurring words in the cache block.
- **COFAE :** COMF augmented with Adaptive Encoding to get more reduction in bit-flips.
- **SWEL-COFAE :** Combination of COFAE with stride-based wear leveling, which helps to shift the write pressure to the middle portion of memory lines.

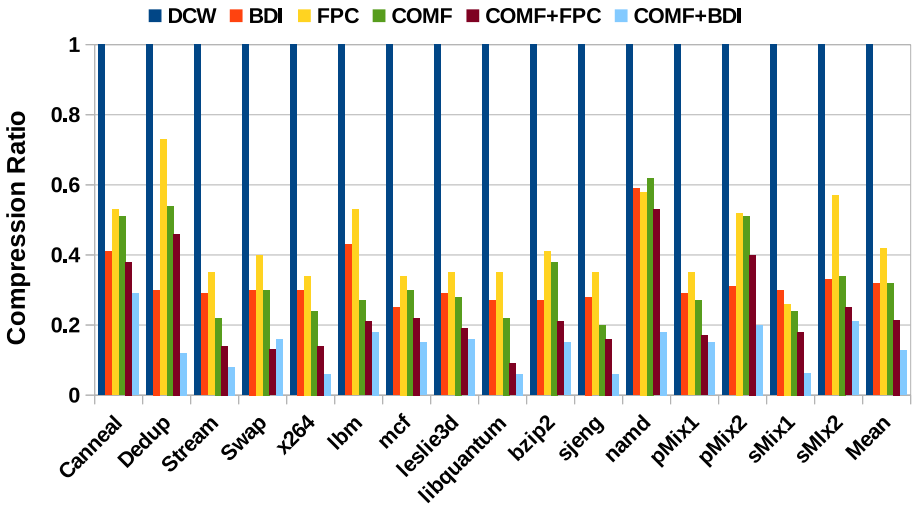


Figure 4.10: *Compression Ratio for BDI, FPC, COMF and Dual-phase (COMF+FPC, COMF+BDI) (Less means more compression, Lesser is better)*

4.4.1 Results and Analysis

We present results for the metrics: bit-flips, energy, lifetime, Cycles Per Instruction (CPI), and Average Memory Access Time (AMAT). All the results are normalized over DCW.

4.4.1.1 Compression Ratio (CR)

Figure 4.10 shows the CR of BDI, FPC, COMF, and dual-phase compression (COMF+FPC, COMF+BDI) for various benchmarks (we have given formal definition of Compression ratio and Coverage in Section 2.3.3). It is evident from Figure 4.10 that we get a fairly good CR for all benchmarks. Certain benchmarks like Streamcluster, Swaptions, x264 have an impressive CR due to the high percentage of repeated words present in the cache blocks of these benchmarks (cf. Figure 4.2). We experimentally found that, the CR of BDI=0.30, FPC = 0.40, COMF = 0.31, and COMF+FPC=0.20, COMF+BDI=0.11 while for DCW it is 1 since it does not use compression. Note that a lower CR indicates higher compression.

COMF outperforms FPC due to a better CR. BDI and COMF have similar CRs; however, BDI has lesser coverage [51]; meaning it is effective on a relatively smaller percentage of blocks compared to COMF. Therefore, overall, COMF outperforms the existing methods.

Bit-flips and energy consumption mainly depend on the number of bits written in PCM. Therefore, the effect of wear leveling is not visible in the evaluation of bit-flips and energy calculation. Hence, we have not shown the results of E1 and SWEL-COFAE for energy and bit-flips (as they are the same as FPC, COFAE, respectively).

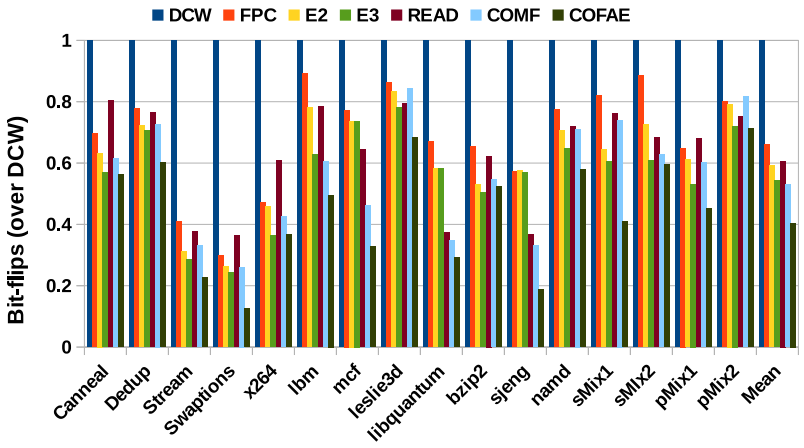


Figure 4.11: *Normalized Bit-flips over DCW (Less is better)*

4.4.1.2 Effect on Bit-flips and Energy Consumption

Compression reduces the number of bits written in the PCM memory, thereby saving a lot of bit-flips. The reduction in bit-flips has a one-to-one correspondence with the reduction in energy consumption and enhancement of PCM lifetime. Figure 4.11 shows normalized bit-flips of various techniques over DCW. On average, the reduction in bit-flips shown by FPC, E2, E3, READ, COMF, and COFAE are 34%, 41%, 46%, 39%, 47% and 59% respectively.

$$\bar{k}_{COFAE} = CR \times \bar{k}_{FNW} \tag{4.3}$$

Equation 4.3 shows the average bit-flips [1] when adaptive encoding, COFAE, is applied to the compressed blocks: Since, $CR < 1$, therefore, $\bar{k}_{COFAE} < \bar{k}_{FNW}$. As per [43], we have $\bar{k}_{FNW} < \bar{k}_{DCW}$. Therefore, we can conclude that $\bar{k}_{COFAE} < \bar{k}_{FNW} < \bar{k}_{DCW}$. Similarly, worst case bit-flips for DCW and FNW are N and $N/2$, respectively. Whereas, the worst case bit-flips in COFAE ($= CR \times (N/2)$), which is scaled by CR is clearly lesser than DCW and FNW.

COMF outperforms the other three compression-based techniques FPC, E2, and BDI/FPC-based E3. Improvement over FPC and E2 is mainly because of its lower CR, leading to reduced size blocks, which in turn reduces the amount of data written to the PCM and reduces bit-flips. E3 compresses the blocks with BDI or FPC based on the resultant size of the compressed block. Although BDI shows low CR, its low coverage forces most of the incoming blocks to be compressed using FPC (which has high CR). As a result, our

[1] Average bit-flips in a memory method m is denoted as $\bar{k}_m$, where m can be DCW, FNW and COFAE. Note that $\bar{k}_m = \sum_{i=1}^{N} ip_i$, where, p_i is the probability of i bit-flips. As per [43].

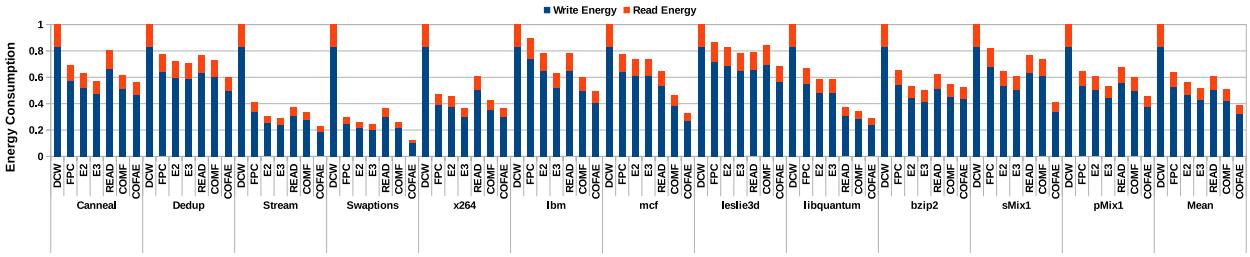


Figure 4.12: *Normalized Energy over DCW (Lesser is better)*

technique is able to perform better than E3. On the other hand, READ encodes the blocks adaptively without compression by avoiding the writing of the clean words. Therefore, it cannot reduce the amount of data written to PCM, leading to poor performance compared to FPC and COMF. Finally, COFAE attains finer encoding granularity by adaptively encoding the compressed blocks produced by COMF. It helps in reducing bit-flips to a higher degree.

For PCM-based main memories, write energy plays a dominant role in the overall energy consumption. We have given the formula for calculating energy consumption in PCM in Section 2.3.2. Figure 4.12 shows the breakup of read and write energy for PCM, which clearly demonstrates the dominance of write energy over read energy (On average, write energy is 5x times more than read energy). Since, the techniques that reduce bit-flips also reduce write energy consumption; therefore they play a significant role in reducing overall energy consumption in PCM. As discussed above, COMF/COFAE reduce bit-flips considerably. It helps in reducing energy consumption. Figure 4.12 shows the normalized energy consumption by existing techniques FPC, E2, E3, READ, and proposed techniques COMF and COFAE over DCW. The reductions in energy consumption (over DCW, for representative benchmarks) by FPC, E2, E3, READ, COMF, COFAE are 36%, 44%, 48% 39% 48% and 61% respectively.

It is also evident from Figures 4.11 and 4.12 that the benchmarks having more compressibility (i.e., higher word locality) like x264, swaptions, streamcluster show more reduction in bit-flips. It can be observed from Table 4.4 that more reduction in bit-flips and energy consumption[1] can be obtained for the highly compressible benchmarks (Rows 4, 5, 6) compared to the benchmarks showing lesser compressibility (Rows 1, 2, 3).

4.4.1.3 Effect on Lifetime

PCM-based main memories have limited write endurance, i.e., they can withstand only a limited number of writes before wearing out completely. Similar to Non-volatile caches, the lifetime of PCM can be defined in two different ways: **Raw Lifetime** and **Error-Tolerant**

[1]Measured as percentage reduction compared to DCW.

Table 4.4: *Relationship between Compressibility and Reduction in Bit-flips, Energy, CR : Compression Ratio, WL : Word Locality*

		CR	WL	%Red. in Bit flips	%Red in Energy	Row No
Less Compress	Canneal	0.51	56.25	40%	29%	1
	Dedup	0.54	62.5	28%	15%	2
	namd	0.62	43.75	42%	40%	3
Highly Compress	Stream	0.22	75	67%	70%	4
	Swap	0.3	81.25	74%	75%	5
	x264	0.24	75	60%	65%	6

Table 4.5: *Average Bit-flip Distribution at Different Bit Positions in a Memory Line after applying Stride-based Wear Leveling*

	0 -99	100 -199	200 -299	300 -399	400 -512	Std. Dev (σ)
Dedup	0.21	0.18	0.20	0.21	0.20	0.012
x264	0.21	0.20	0.19	0.21	0.19	0.01
Fluid	0.18	0.20	0.22	0.19	0.21	0.015
Canneal	0.19	0.20	0.20	0.19	0.22	0.012
Stream	0.22	0.21	0.18	0.19	0.20	0.015
Mean	**0.20**	**0.19**	**0.19**	**0.20**	**0.20**	**0.005**

Lifetime. In this work, we focus on improving the raw lifetime since it is the base for error-tolerant lifetime [74] (For details, kindly refer to Section 2.3.1).

COMF enhances the PCM lifetime by reducing bit-flips in the PCM cells. The reduction in bit-flips is more for COMF than FPC, E2, E3, and READ (explained above). The addition of adaptive encoding (COFAE) and stride-based wear leveling to COFAE (SWEL-COFAE) catalyzes the lifetime improvement to a further extent. Compared to the orientation-based wear leveling (cf. results from Table 4.1), the stride-based wear leveling shifts the bit-flips towards the less write activity-prone middle cells of the memory lines, which gives a better balance of bit-flips within the PCM cells. As shown in Table 4.5, the average normalized bit-flips in the bit positions 0-99, 100-199, 200-299, 300-399 and 400-512 bit are 0.20, 0.19, 0.19, 0.20 and 0.20 respectively (with σ_{avg}=0.005). An achievement of a fairly uniform distribution compared to the previous uneven distribution : 0.25, 0.17, 0.14, 0.16, 0.26 (with σ_{avg}=0.048, from Table 4.1). In particular, Stride-based wear leveling achieves a reduction of 89% in average standard deviation (σ_{avg}) of bit-flips across the bit positions over orientation-based wear leveling. This uniformity in the bit-flips reduces maximum writes to the individual cells and leads to an eventual improvement in the PCM lifetime. Table 4.6

Table 4.6: *Bit-flips, IntraV and Lifetime improvement of Stride-based over Orientation-based Wear Leveling*

Benchmarks	%Reduction in peak bit-flips	% Reduction in IntraV	%Lifetime Improvement
Canneal	32	18	12
Dedup	20	40	22
x264	14	25	27
Fluid	39	42	32
Stream	10	22	6
Mean	**20**	**30**	**17**

shows the percentage reduction in peak bit-flips and IntraV [1]and the eventual improvement in lifetime by stride-based over the orientation-based approach. On average, the reductions in maximum writes and IntraV are 20% and 30%, respectively, whereas the improvement in lifetime is 17%.

SWEL-COFAE outperforms DEUCE-based Horizontal Wear Leveling (HWL) with an improvement of 11% over HWL. The rotation amount[2] defined in HWL is same for all the blocks, and it changes very slowly depending on a *Start* pointer. Therefore, it can not evenly distribute the bit-flips within the blocks that face frequent writes. On the contrary, SWEL-COFAE uses a more localized policy where rotation amount is defined for individual blocks, which changes depending on the number of writes issued to a particular block. Therefore, the bit-flips within the memory lines can be more evenly distributed by SWEL-COFAE than HWL.

Finally, the improvement in lifetime for E1, E2, E3, READ, HWL and SWEL-COFAE over DCW are 50%, 39%, 57%, 58%, 81% and 101% respectively (shown in Figure 4.13). A variant of application of our proposal could be to perform wear leveling (i.e., position of block writing) before encoding. This gives similar results to performing encoding followed

[1]Defined in [74] for Intra-set write variation LLC and re-written for Intra-Block write variation (IntraV) for NVM in [111], where BFi,j is the write count of cell j in block i, and BF_{avg} is the average bit-flips count and N is the total number of blocks.

$$IntraV = \frac{1}{BF_{avg}.N} \sum_{i=1}^{N} \sqrt{\frac{\sum_{j=1}^{512} \left(BF_{i,j} - \sum_{j=1}^{512} BF_{i,j}/512 \right)^2}{511}} \tag{4.4}$$

[2]Rotaion amount is determined using pointers *Start* and *Gap* that point to a starting memory line and a dummy *gap* line, respectively. *Gap* pointer along with the *gap* line moves to the neighboring line after a certain number of writes to the bank. *Start* is incremented when *Gap* pointer makes a complete tour of all the memory lines in the bank. *Rotation amount* = $Start'$ *%Bits in line*, where $Start'$ equals (*Start*+1) if the *Gap* has already crossed the line, otherwise *Start* is equal to *Start*.

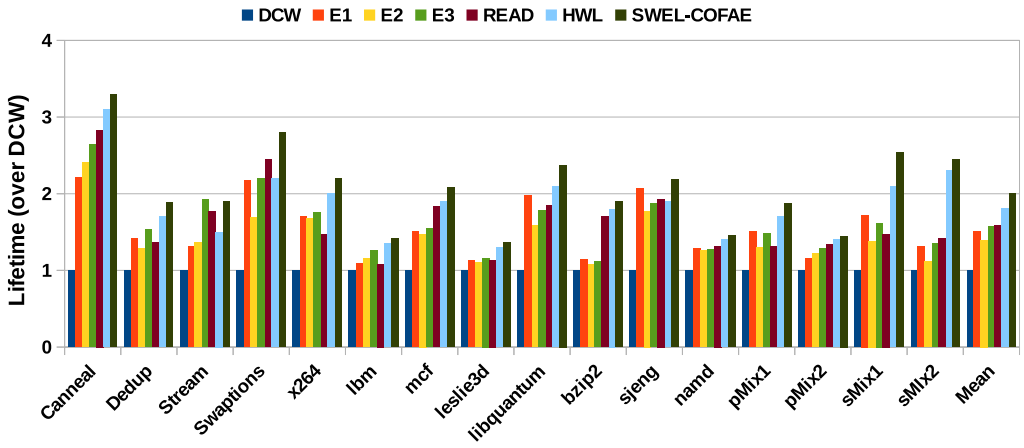


Figure 4.13: *Normalized Lifetime over DCW (More is better)*

by wear leveling, as the main impact is from size reduction due to block compression.

4.4.1.4 Effect on System Performance

Cycles Per Instruction (CPI)

Lesser CPI indicates an improvement in the system performance. Compression helps in quicker delivery and writing of compressed data on read and writes, respectively. Therefore, the large main memory read and write latencies can be very well reduced with compression. Results related to performance are shown in Figure 4.14. Since COMF achieves a better CR than FPC, therefore savings in main memory read and write cycles due to compression will be more for COMF/COFAE. This minimization of write bits improves the access speed by reducing the response time, which in turn improves the performance. On the other hand, in the case of READ, the reduction in bit-flips is not as high as COMF/COFAE. Therefore, the improvement shown by READ is lesser compared to COMF/COFAE. BDI/FPC-based compression in E3 brings less benefit as the selection done between the two methods depends on the CR as well as coverage. On average, the techniques FPC, E2, E3, READ, COMF, and COFAE improve CPI by 18%, 20%, 21%, 19%, 21%, and 25% over DCW, respectively. In particular, COFAE uplifts the performance by 4% over COMF.

Benchmarks having high write intensity (lbm, mcf, leslie3d, sjeng) show higher improvements in CPI than the benchmarks with low write intensity (dedup, stream, swap, x264, bzip2). COFAE reduces the write service time due to high compression, leading to improved CPI for write-intense benchmarks. Furthermore, COFAE also shows high improvement in performance for benchmarks (mcf, bzip2, leslie3d) where the reads are much more than the writes. This is due to the reduced read service time of the compressed blocks stored in PCM. In particular, the performance improvement shown by COFAE for benchmarks with

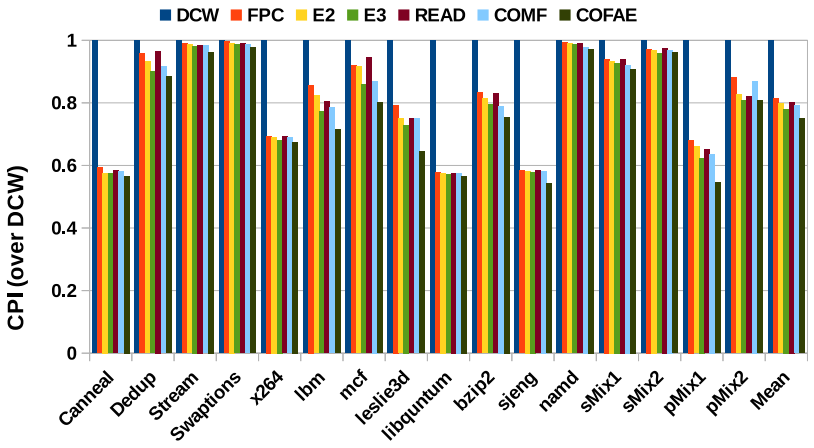


Figure 4.14: *Normalized CPI over DCW (Lesser is better)*

high write intensity, low write intensity and when reads are much more than writes are 31%, 14%, and 22%, respectively. High degree of compression, decent coverage of COMF, and adaptive encoding make COFAE robust enough to perform well in all these scenarios compared to other compression-based techniques where the degree of compression (e.g., FPC) or coverage (e.g., BDI) is low.

Average Memory Access Time (AMAT)
AMAT gives the average estimated time taken to deliver the data items from the memory hierarchy. Compressed blocks stored in the PCM arrays can be delivered quickly, reducing the LLC miss penalty. COMF outperforms FPC and BDI in reducing AMAT. Compared to FPC, COMF produces smaller blocks, owing to its smaller CR. On the other hand, although BDI shows comparable CR with COMF, the percentage of blocks compressed by BDI is lesser.

As shown in Figure 4.15, shows AMAT for BDI, FPC, and COMF normalized over DCW. Reduction in AMAT by BDI, FPC, and COMF is 42%, 57%, and 64% over DCW, respectively. Comparison of AMAT is shown only for compression techniques because the reduction in block size affects AMAT, whereas the encoding techniques do not reduce block size, and hence we do not show AMAT results for COFAE and other methods.

4.4.2 Comparative Analysis

In addition to the above results, we also provide results related to varying compression threshold, number of encoding tag bits and cache block size for better comprehension of our proposed technique.

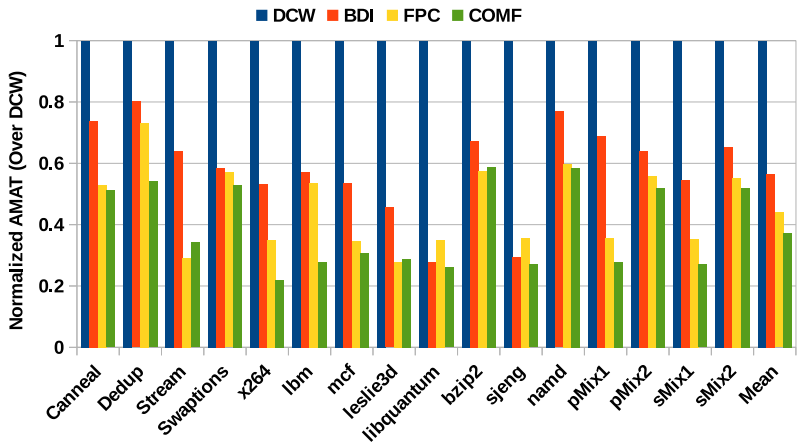


Figure 4.15: *Normalized AMAT Over DCW (Lesser is better)*

Table 4.7: *Effect of Varying Compression Threshold on Compression Ratio (CR) and Latency Overhead; Benchmarks are classified based on write intensity as Low (L), Medium (M), High (H)*

	th= 4		th= 8		th= 12	
	CR	%Latency Overhead	CR	%Latency Overhead	CR	%Latency Overhead
Canneal (H)	0.38	3.9	0.51	2.4	0.6	1.12
Dedup (M)	0.46	3.7	0.54	3.1	0.73	2.2
Freqmine (L)	0.45	3.1	0.52	2.3	0.56	0.88
Fluid (M)	0.25	3.6	0.35	2.1	0.4	1.54
Stream (L)	0.15	3.9	0.22	3.7	0.35	3.3
Swaptions (L)	0.13	4.6	0.3	4.4	0.4	3.9
x264 (M)	0.15	3.8	0.24	3.5	0.34	3.3
Mean	**0.25**	**3.8**	**0.37**	**2.9**	**0.47**	**2**

Table 4.8: *Effect of different number of Tag Bits on Storage Overhead and Bit-flip reduction for Non-Adaptive Vs. Adaptive Encoding*

Tag Bits	%Storage Overhead	Non-Adaptive Encoding-FNW		Adaptive Encoding -COFAE	
		Gran	%Red. In Bitflips	Gran	%Red. in Bitflips
8	1.56	64	44	25	51
16	3.12	32	49	12	58
32	6.25	16	56	6	66

4.4.2.1 Compression Threshold of COMF

COMF uses a threshold (th) to decide whether a block is compressible or not. If the frequency of the MFW in a block is more than th, the block is compressed. Otherwise, the block is written in PCM without compression. However, there exists a trade-off between the CR and latency overhead associated with the compression/decompression. For lower values of th, the number of blocks compressed by COMF increases, which leads to a smaller CR. A smaller CR indicates that the average size of the cache blocks becomes smaller after compression, which helps in reducing the bit-flips and energy consumption in PCM. However, the delay associated with compression/decompression increases with the increase in the number of compressed blocks. It is evident from Table 4.7 that as th increases, the latency overhead decreases but at the cost of an increased CR. On average, for $th = 4, 8$, and 12, the CR are 0.25, 0.37, and 0.47, respectively, whereas the percentage latency overhead are 3.8%, 2.9%, and 2%, respectively. For $th = 4$, CR is lowest but, latency overhead is highest; while for $th = 12$, the scenario gets reversed with lowest latency overhead and highest CR. Therefore, we decided to take $th = 8$ for our experiments that best balances the inherent trade-off between CR and latency overhead.

4.4.2.2 Comparison of Non-Adaptive Vs Adaptive Encoding for Different Number of Tag Bits

Adaptive Encoding on the compressed blocks leads to finer granularity compared to its Non-Adaptive counterpart. In order to give an average estimate of the Encoding Granularity achieved using Adaptive Encoding, we define a term called *Effective Granularity* ($Gran_{eff}$) as follows:

$$Gran_{eff} = \frac{b_{uc}}{T} \times CR = Gran_{Fixed} \times CR \tag{4.5}$$

Table 4.9: *Compression Ratio (CR) Vs Compression/Decompression Latency Overhead (OH) for Different Cache Block Sizes*

Block Size=64 B			Block Size=128 B		
	CR	OH		CR	OH
th=4	0.25	3.8	th=12	0.22	5.7
th=8	0.36	2.9	th=16	0.34	4.8
th=12	0.46	2	th=20	0.44	3.6

Table 4.10: *Non-adaptive-FNW Vs Adaptive-COFAE for Varying Cache Block Size;* ***%RB :*** *% Bit-flip reduction over DCW ;* ***SO :*** *Storage Overhead;* G_F *: Fixed Encoding Granularity* ($Gran_{Fixed}$) *;* G_e *: Effective Encoding Granularity* ($Gran_{eff}$)

Block Size=64 bytes						Block Size=128 bytes					
		FNW		COFAE				FNW		COFAE	
	SO	G_F	%RB	G_e	%RB		SO	G_F	%RB	G_e	%RB
Tag=8 bits	1.56%	64	44	25	51	Tag=16 bits	1.56%	64	38	24	46
Tag=16 bits	3.125%	32	49	12	58	Tag=32 bits	3.125%	32	42	14	52
Tag=32 bits	6.25%	16	56	6	66	Tag=64 bits	6.25%	16	48	8	62

where CR is the Compression Ratio (b_{uc} and T are defined in equation 4.1, CR is calculated using Equation 2.5). As $Gran_{Fixed} = b_{uc}/T$ (as defined in Section 2.2.1), we can also write the equation using $Gran_{Fixed}$.

The numerator ($b_{uc} \times CR$) in the Equation 4.5 gives the average number of bits in a compressed block. Therefore, $Gran_{eff}$ essentially denotes the average number of bits in the compressed blocks that are represented by one tag bit. In other words, $Gran_{eff}$ gives a representational value of the average *Adaptive Encoding Granularity* to gauge the effectiveness of the *Adaptive Encoding* utilized by COFAE.

Table 4.8 shows the effect of different number of tag bits on storage overhead and bit-flip reduction for Adaptive Vs. Non-Adaptive Encoding. It is evident from the table that using the same number of tag bits, $Gran_{eff}$ achieved by adaptive encoding is finer compared to the $Gran_{Fixed}$. As a result, the reduction in bit-flips is more for our proposed Adaptive Encoding based scheme-COFAE than the Non-Adaptive Encoding-FNW. It is also interesting to observe from Table 4.8 that the reduction in bit-flips achieved by COFAE is better than the reduction achieved by FNW even after using twice the number of tag bits (compared to COFAE). For example, the reduction in bit-flips by COFAE is 51% using only 8 tag bits (i.e., storage overhead due to tag bits = 1.56%), whereas FNW is able to achieve a similar reduction of 49% by using 16 tag bits (storage overhead =3.12%). It establishes the fact that our Adaptive Encoding can achieve much more bit-flip reduction than Non-Adaptive Encoding at a relatively lower storage overhead.

Table 4.11: *Percentage Reduction in Bit-flips and Energy on Supplementing COFAE with FPC and SAE*

Techniques	%Reduction in Bit Flips	%Reduction in Energy
FPC	34	34
E2	41	42
E3	47	47
READ	35	35
COMF	47	48
COMF+SAE	50	52
COFAE	59	61
COFAE+SAE	63	64
COFAE+FPC	65	67

Table 4.12: *Effect on Lifetime on Supplementing SWEL-COFAE with FPC*

Technique	Lifetime
DCW	1
E1	1.52
E2	1.40
E3	1.61
READ	1.56
SWEL-COFAE	2.01
SWEL-COFAE+FPC	2.28

4.4.2.3 Change in the Cache block size

Effect on Compression Threshold : Table 4.9 shows the relation between CR and compression/decompression latency overhead for different block sizes. High compression thresholds are needed for 128 byte blocks in order to maintain a similar CR and latency overhead as that of 64 byte blocks. It is also evident from the table that threshold (th=16) equal to half the number of words (32 words in 128 byte block) in the block balances the trade-off between CR and latency overhead (discussed in Section 4.4.2.1) in the best possible manner, similar to 64 byte block where the threshold is set to 8. Higher latency overhead for 128 byte blocks is due to high compression/decompression latency involved in compressing/decompressing large size blocks.

Effect on Tag Bits used for Encoding : Table 4.10 shows the reduction in bit-flips by Non-Adaptive-FNW Vs. Adaptive-COFAE for 64 byte and 128 byte blocks. It can be seen from the table that the number of tag bits used should be double for 128 byte blocks compared to 64 byte blocks for similar storage overhead, effective encoding granularity ($Gran_{eff}$), and reduction in bit-flips. Similar to 64 byte blocks, COFAE outperforms FNW for 128 byte blocks also in reducing bit-flips, since the degree of compression gets maintained for larger blocks also, which results in finer granularity encoding for compressed blocks.

4.4.3 Effect of supplementing COMF/COFAE with FPC and SAE

COMF compresses the cache blocks at a word-level granularity. However, it can be easily integrated with the compression techniques that operate on the bit-level granularity, making

COMF a dual-phase compression scheme [1]. We integrate FPC [89], a popular bit-level compression technique on top of COMF as the second phase of compression. Out of many bit-level compression techniques, we have selected FPC because of its low implementation and storage overhead. FPC compresses the words within a cache block based on the data patterns stored in a table.

When COFAE is backed by FPC, it leads to more reduction in bit-flips, energy consumption, and more longevity of PCM main memory. Tables 4.11 and 4.12 show the effect of using FPC as a second phase compression. Reduction in bit-flips and energy consumption when FPC is augmented with our proposed scheme are 65% and 67% over DCW, respectively, while improvement in lifetime is 128%. More specifically, SWEL-COFAE, on integration with FPC achieves reduction of 47% in bit-flip, 37% in energy, and lifetime improvement of 49% over FPC-only technique.

Sequential flips Aware Encoding (SAE) [44] technique can also be integrated on top of COMF and COFAE to reduce bit-flips further. SAE helps in reducing the bit-flips in the tag area by selecting the appropriate encoding granularity, which otherwise may be high when fine granularity encoding is applied. The results of augmenting SAE on top of COMF and COFAE are shown in Table 4.11. Both COMF and COFAE show improvement in effectiveness when SAE is augmented with them.

4.4.4 Overhead Analysis

Storage Overhead: SWEL-COFAE needs a total of 19 bits of meta-data (1 compression bit for COMF, 16 bits for encoding tags, and 2 bits (i.e., 1 orientation bit and 1 stride bit) for stride-based wear leveling) per cache block. Therefore, the storage overhead of SWEL-COFAE is $\frac{19}{512} * 100\% = 3.7\%$.

Hardware Overhead Analysis: We implemented the design of COMF Compressor/ Decompressor module in synthesizable Verilog Hardware Description Language (HDL). We then performed placement aware logic synthesis in Genus Synthesis Solution from Cadence (15nm technology node). The obtained latency, power, and area values of Compressor/Decompressor are shown in Table 4.13. Table 4.14 shows the area, power and latency values in two technology nodes (15nm, 45nm) for a better comparison. The latencies are very small compared to the large main memory latency, so they have little impact on performance. The compressor/decompressor hardware occupies 0.16% of a standard memory controller like the one mentioned in [135, 136], at 15nm technology node. Finally, as COFAE uses adaptive encoding compared to static encoding of FNW, it needs 1.16x and 1.19x more area and power, respectively, compared to FNW.

[1] The rationale behind supplementing FPC with COMF is to show the ease of integration of COMF with any existing compression technique, although COMF alone provides nice compression.

Table 4.13: *Latency, Area and Power Analysis of the COMF hardware*

	Compression	Decompression
Latency (ns)	3.5	2.65
Area (μ^2)	2252.46	75
Total Power (mW)	0.788	0.062

Table 4.14: *Latency, Area and Power Analysis of the COMF hardware for 15nm and 45nm nodes*

	COMF @ 45nm	COMF @ 15nm
Latency (ns)	8.43, 6.43	3.5, 2.65
Area (μ^2)	13500	2252.46
Power (mW)	5.42	0.788

Table 4.15: *Comparison of Storage Overhead, Normalized Bit-flips, Energy and Lifetime over DCW*

Technique	Overhead (%)	Bit-flips	Energy	Lifetime
DCW	0	1	1	1
E1	6.25	0.66	0.65	1.52
E2	0.2	0.59	0.58	1.40
E3	0.4	0.53	0.54	1.61
READ	7.8	0.65	0.64	1.56
SWEL-COFAE	3.7	0.42	0.41	2.01
SWEL-COFAE+FPC	3.9	0.35	0.33	2.28

4.4.5 Comparison of Capacity overhead, Bit-flips, Energy and Lifetime

Table 4.15 shows a comparative analysis for DCW, E1, E2, E3, READ, SWEL-COFAE, and when dual-phase compression is applied by supplementing COFAE with FPC[89] (SWEL-COFAE+FPC). SWEL-COFAE alone shows improved results in terms of bit-flips, energy, and lifetime while incurring comparable/better capacity overhead (3.7%) than other methods. Combining COFAE with FPC and wear leveling (SWEL-COFAE+FPC), we get even better improvements in terms of bit-flips, energy, lifetime, and performance. The increase in storage overhead when our proposed technique is supplemented with FPC is very less (an increase from 3.7% to 3.9%).

4.4.6 Discussion

FPC-based techniques like [49, 50] show poor compressibility and high meta-data overhead. On the other hand, COMF provides more compressibility at a relatively lower meta-data overhead, leading to reduction in bit-flips in PCM. COFAE reduces bit-flips to a further extent by associating Adaptive Encoding on the COMF compressed blocks. Advantages of COFAE with FNW[43] is discussed in Section 4.3.2.2. COMF/COFAE can be made more effective by augmenting SAE[44] on top of them. Finally, SWEL-COFAE that integrates COFAE with stride-based wear leveling turns out to be more effective in balancing the intra-line bit-flips than the state-of-the-art wear leveling techiques RNW [97] and HWL [39] (comparison between SWEL-COFAE and HWL is provided in Section 4.4.1.3). SWEL-COFAE also provides better uniformity of bit-flips than RNW as RNW leads to an increased write pressure in one half of memory line where the compressed data gets written.

4.5 Summary

Non-volatile memories are promising candidates for constructing high-density and energy-efficient main memories. However, downsides like limited write endurance and high write energy thwart their scope of adoption. In this chapter, we propose a word-level compression technique called COMF to reduce bit-flips in PCM-based main memories. COMF is augmented with an adaptive granularity-based encoding that reduces bit-flips to a greater extent. The combined technique of COMF and Adaptive Encoding is termed as COFAE. In order to improve the PCM lifetime further, we propose an intra-line wear-leveling technique with COFAE that distributes the intra-line bit-flips by periodically changing the orientation of writing and shifting the bit flip pressure towards less write activity prone middle cells. The integrated solution of COMF, Adaptive Encoding, and Stride-based wear leveling is termed as SWEL-COFAE.

Experimental results show that SWEL-COFAE reduces bit-flips by 59%, 35%, 28%, 36% ; reduces energy consumption by 61%, 19%, 14%, 22% and improves lifetime by 101%, 22%, 27%, 19% over DCW and three state-of-the-art techniques E1 [49], E2 [50] and READ [44], respectively, while incurring a smaller capacity overhead of 3.7%. Thus, seeking newer avenues of data compression, paired with wear leveling, can assist in the longevity of PCM-based main memories.

5

Pop-Crypt : Reducing Encryption Overhead in NVMs by Identification of Popular Words

The non-volatility feature of the NVMs may lead to stealing of the sensitive data stored in NVMs due to their prolonged data retention. Memory encryption turns out to be a viable option to provide data security. However, the existing encryption techniques, on account of their diffusion property, increase the number of bit-flips in the NVM cells, thus leading to their early wear out. Therefore, security and lifetime issues of the NVMs are difficult to go hand in hand. In this chapter, we identify words that are repeated across several memory blocks and term them as popular words. The proposal is to avoid the encryption of the popular words by maintaining them in a reference table. In our proposal, Pop-Crypt, every block to be written to the memory gets partially encrypted in which the popular words are replaced with pointers to the reference table, and other words get encrypted. The partially encrypted blocks reduce the number of bit-flips in PCM, thereby improving its lifetime significantly. Experimental results show that Pop-Crypt considerably improves lifetime, energy consumption, and system performance over baseline and state-of-the-art techniques.

5.1 Introduction

The non-volatility feature of the NVMs offers help in dealing with the power/system failures, check pointing improvement [137], and reducing applications start-up [138]. However, due

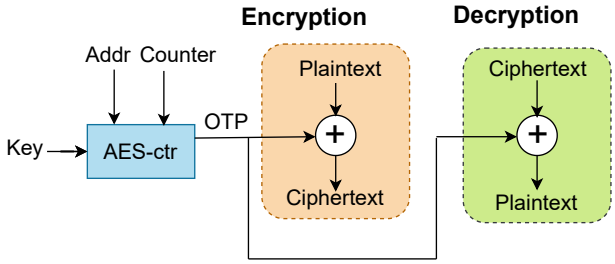


Figure 5.1: *AES-based Counter Mode Encryption*

to non-volatility, the data in NVMs remain persistent even after the system is powered down. Therefore, an attacker having physical access to an NVM DIMM can easily stream out confidential data. This type of attack is called Stolen DIMM attack [39, 40]. Encryption offers a firm security guarantee to the data against such attacks. AES-based counter mode encryption (CME) [39, 113] has been adopted in NVM premises due to its low decryption penalty. CME uses a counter, a secret key (stored securely in the processor-side memory controller), and the line address to generate a One Time Pad (OTP). The OTP is XORed with the plain text/cipher text to obtain the cipher text/plain text, as shown in Figure 5.1.

However, encryption shows diffusion property [39], i.e., changing even a small number of bits in the plain text results in enormous bit-flips in the generated cipher text. These write-activities tend to shorten the NVM lifetime severely. Unfortunately, the commonly used techniques like DCW, FNW to reduce bit-flips do not show impressive outcomes for encrypted memories. Therefore, there is a strong need to develop solutions that control the deterioration of the lifetime of encrypted NVMs due to the diffusion property of the encryption techniques. Existing techniques DEUCE [39] and SECRET [40] partially encrypt the incoming cache lines by avoiding the encryption of the unmodified words. However, these methods need occasional re-encryption of the whole cache blocks, which increases bit-flips. In this contribution, we take a different approach to avoid encryption of the frequently occurring words that are repeated across the cache blocks. This is done by maintaining a table of such words and writing the index for the word from this table in the respective word position instead of encrypting it while writing the block in PCM. Our experimental results confirm that many words present in the incoming cache lines to PCM are repeated with a high frequency across the cache lines. We call these words as *Popular Words* and collect them in a table called Popular Word Table (PWT) inside the memory controller for future reference. The encryption of these popular words can be skipped during block encryption, which reduces bit-flips in PCM. Since the set of popular words varies across applications, a period known as the training period is explicitly dedicated to collect the popular words in the PWT. The PWT is updated at runtime to contain the collection of the most popular words during the entire period of execution.

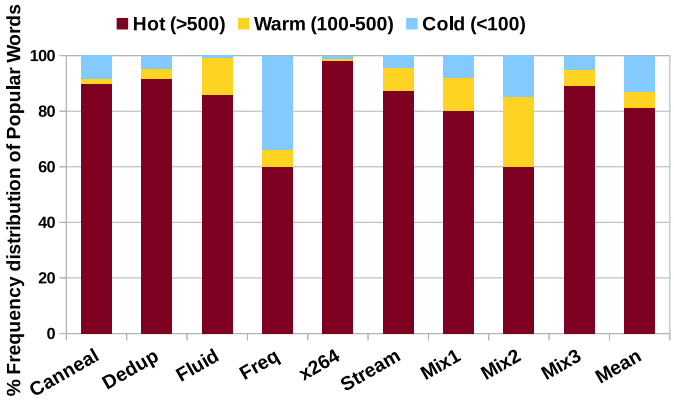


Figure 5.2: *Percentage Frequency Distribution of the Popular Words*

5.2 Chapter Overview

5.2.1 Contributions of the Chapter

The main contributions of the chapter are as follows :

- We propose a technique called Pop-Crypt that avoids the encryption of the frequently occurring words (popular words) in the incoming cache lines. It reduces bit-flips in the PCM cells significantly.
- We have provided necessary sensitivity analysis on determining the suitable size of the PWT, selecting the appropriate word length, and the length of training interval of Pop-Crypt.

5.2.2 Chapter Organization

The rest of the chapter is organized as follows. The motivation is presented in Section 5.3. The proposed methodology is discussed in Section 5.4. Section 5.5 illustrates the experimental evaluation, followed by summary in Section 5.6.

5.3 Motivation

There exists a high degree of word-level redundancy in the data blocks that get written to the main memory. The data blocks are modified by the applications and travel through the memory hierarchy. In particular, these data blocks are the cache lines evicted by the Last Level Cache (LLC). Our experimental results confirm the existence of some highly frequent words across these cache lines coming to PCM. We divide the words as hot, warm, and

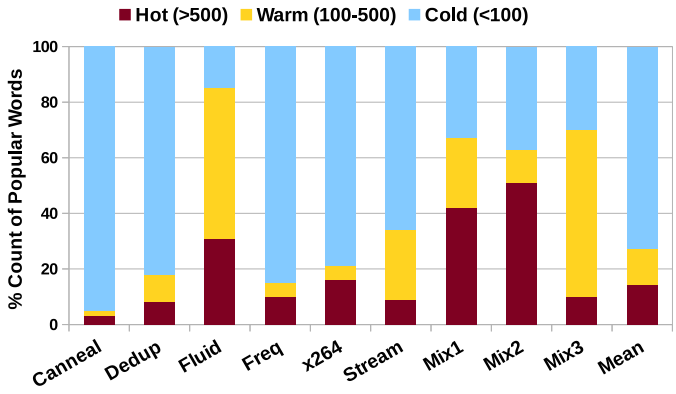


Figure 5.3: *Percentage Count of Popular Words*

cold if their frequency of occurrence is more than 500, between 100-500, and less than 100, respectively, during application execution. Note that we determine these thresholds based on the frequency of the words across several benchmarks. Figures 5.2 and 5.3 show the percentage distribution of frequency and counts of the hot, warm, and cold words, respectively. It is evident from Figure 5.2 that the frequency of hot words dominates the frequency of warm and cold words (On average, 82%, 6%, and 12%, respectively). Interestingly, the count of hot words is very less compared to the warm and cold words, as can be seen clearly in Figure 5.3 (On average, 14%, 12%, and 74%, respectively). These figures show that the words that occur with high frequency are very few. We term such highly frequent words as popular words. These popular words can be maintained in a small table for future reference. When the incoming cache blocks are encrypted before writing to PCM, the encryption of the popular words inside the cache blocks can be skipped[1]. When these Partially Encrypted Cache Blocks (PEB) are written to PCM, a lot of bit-flips are saved, thereby enhancing the lifetime of PCM.

5.4 Proposed Methodology

In this section, we will first explain the architecture of our proposed methodology. Then, we will describe in detail the working principle of our proposed technique : Pop-Crypt.

[1]Pop-crypt partially encrypts the popular words present within the incoming cache blocks. However, the non-popular words remain encrypted in their previous encrypted state. Hence, when an adversary scans the entire block, he will still find the entire block in encrypted state.

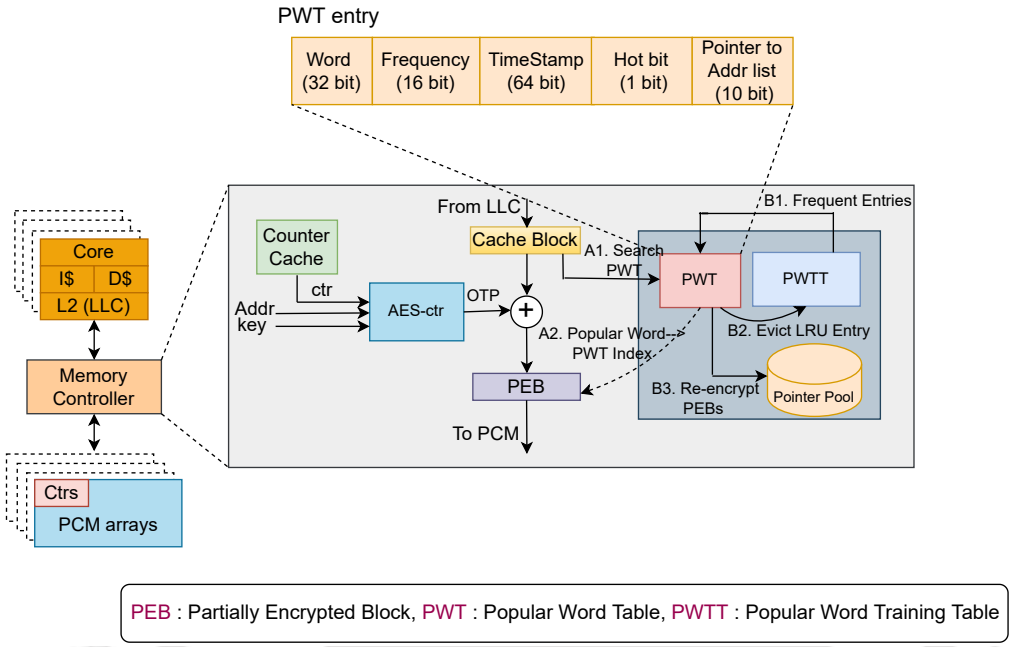


Figure 5.4: *Architecture of the Proposed Technique : Pop-Crypt*

5.4.1 Architecture

Figure 5.4 shows the architectural view of Pop-Crypt. We extend the memory controller by incorporating an AES engine for encryption/decryption. A counter cache is maintained to improve performance that stores the counters of the recently accessed blocks. The frequent words are kept in the Popular Word Table (PWT). A Popular Word Training Table (PWTT) is used for collecting the information regarding the newly popular words. A small storage called Pointer Pool keeps the memory addresses of the blocks containing the PWT indices.

Pop-Crypt partially encrypts the incoming cache blocks by skipping the encryption of the popular words in them. The words of an incoming block are searched in the PWT (shown by arrow A1). Encryption of the popular words is skipped by keeping only their PWT index in the respective word position (shown by A2). On the other hand, the non-popular words are encrypted by XORing them with the respective bits of the generated OTP. A Bit Presence Vector (BPV) (dedicating 1 bit per word) is formed to keep track of the popular/non-popular words inside a block. The BPV and PWT indices help in identifying the popular words during the decryption process. PWTT keeps information of the other frequent words that have a high chance of becoming popular in the near future. A new word with high frequency is declared popular and brought to PWT (shown by B1) by replacing the cold Least Recently Used (LRU) entry in the PWT (Shown by B2). The blocks that share the evicted cold word from PWT are re-encrypted using the addresses stored in the pointer pool (shown by B3). We term the blocks that contain the indices of

PWT for popular words and encrypted non-popular words as Partially Encrypted Blocks (PEBs). Since the frequency of the popular words is high, the PEBs contain many words whose encryption are skipped, leading to a reduction in bit-flips in PCM.

5.4.2 Proposed Technique : Pop-Crypt

Pop-Crypt aims to reduce bit-flips in PCM by skipping the encryption of popular words present inside the cache lines. In order to maintain the collection of the popular words, we need certain data structures that get updated at runtime. In this section, we briefly elaborate on these data structures, the process of constructing the PWT, and its use during normal execution.

5.4.2.1 Data structures used

Below we describe the data structures used by our technique : PopCrypt.

(a) Popular Word Table (PWT)
It is a small table with 64 entries [1] containing the information of the most frequently accessed words present in the incoming cache lines to PCM. The various fields of a PWT entry are :

- **Word** : The popular word which is 32 bits in length.
- **Frequency**: It is a 16-bit field that gives the frequency of occurrence of the corresponding word.
- **Timestamp**: The time at which the word last appeared in an incoming cache line to PCM.
- **Hot bit**: 1-bit field that indicates that the word's frequency is more than a predefined threshold. The word is declared as a hot word.
- **Valid bit**: 1-bit field that indicates whether the PWT entry is free (0) or allocated (1).
- **Pointer to Address list**: A 10-bit pointer to the address list associated with cold word entry.

(b) Popular Word Training Table (PWTT)
It collects the information of the newly popular words. The entries have the same fields

[1] We give sensitivity analysis on various sizes of PWT in Section 5.5.2.1

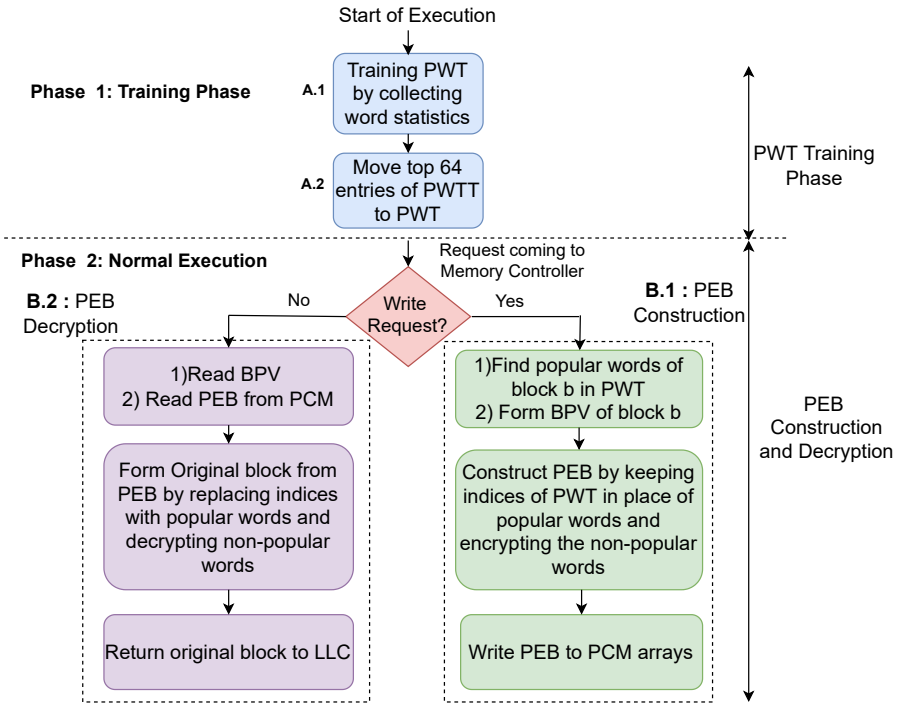


Figure 5.5: *Flowchart of the Working of Pop-Crypt*

(except the pointer to address list field) as the PWT. We keep the size of PWTT (512-entries) sufficiently high relative to PWT to accommodate the information of a large number of newly popular words.

Since the popular words are distinct across the applications, we use an initial training (of length 100M cycles [1]) phase at the beginning of application execution to identify the popular words. Accordingly, we divide the execution of Pop-Crypt into two phases: 1) Training Phase 2) Normal Execution. At the end of the training phase, the popular words are identified, and PWT is updated. However, certain new popular words may get identified beyond the training phase. Our policy keeps updating the PWT at runtime with the help of the PWTT in order to include such newly popular words in the PWT. Figure 5.5 gives the overall working steps of Pop-Crypt.

(c) Pointer Pool

The hot words in the PWT are found in abundance in memory, and we do not keep track of which addresses have used these hot words. However, for cold words, we keep track of the addresses using them. This is done so that in case we need to replace a cold word with another word in the PWT; we will have to re-encrypt the addresses using this cold word.

[1] We give a sensitive analysis in Section 5.5.2.2 on setting the duration of training phase as 100M cycles.

The addresses are maintained as list of pointers. To keep the storage overhead to an optimal value, the storage required for such pointers is maintained as a pool of pointers, called the *Pointer pool*. Whenever a cold word is evicted, the addresses sharing it are re-encrypted, and the pointer storage is freed. These free pointer locations can be used by newer cold words in the PWT.

This process is done in the background without hampering normal read/write operations to reduce its effect on system performance. Also, our experiments reveal that the likelihood of PWT evictions after the training phase is very low (Refer to Section 5.5.2.4 and Table 5.3). When the memory pages are relocated to different memory regions/swapped to disk, the pointer pool is updated by updating the lists associated with the PWT entries to reflect the new memory addresses. Each entry in the pointer pool has the following fields:

1. Free bit : It tells whether the entry contains the address of a memory block (with Free bit=0) or free (Free bit=1).
2. Physical address of the memory block containing a popular word if Free bit=0, else that field is empty.
3. Next Index field that points to the next entry in a list.

5.4.2.2 Steps During Execution

The PWT is constructed by following a training period. Once the training interval is completed, the PWT is used for normal operations. Note that the training using PWTT continues in the background during normal operation to identify newer words (if any) to update the PWT.

1) Training Phase
In this phase, popular words are collected in the PWT using the PWTT (shown as rectangle A.1 in the flowchart in Figure 5.5). For any incoming cache line, its words are searched in the PWTT for a match. On finding a match in an entry in the PWTT, the corresponding *frequency* field of the entry is incremented. On the other hand, if the word is not found in the PWTT, a new entry for the word is created by removing the least frequent entry from the PWTT. At the end of the training phase, PWTT contains the entries of the topmost 512 frequent words. Out of these, the top 64 word entries are copied to the PWT so that it contains the top 64 most frequent words (Rectangle A.2 in Figure 5.5).

Reliability of PWT: Keeping in view the possible loss of the PWT and pointer pool contents during a system power off, their backup is taken (in encrypted form) in PCM after the training phase is over. Any subsequent updates in PWT and pointer pool are also sent to the PCM storage during idle time slots (i.e., when the memory requests are less). Note that PWT gathers popular words of all the applications running concurrently in the system. Our experiments reveal that some highly frequent popular words are shared across

different benchmarks (12%-17% for the workload Mixes) which remain present in the PWT even after the context switches. Also the popular words belonging to different applications change infrequently after the training phase (kindly refer to Figure 5.13 and Section 5.5.2.2). Therefore, context switch of the running applications do not impact the performance of Pop-Crypt. Note that, dictionary compression techniques compress data based on the frequent values present in the dictionary. However, many of such techniques [139, 140] are non-adaptive in nature which build their dictionary statically via profiling or by using a training period for a small duration of time to gather the frequent values. On the other hand, the adaptive approach incurs high overhead as the evicted entry leads to further updates in the data items that refer to the evicted entries [141, 142]. Pop-Crypt adaptively updates the PWT without degrading the performance.

2) Normal Operation

At the beginning of normal operation, the words collected in the PWT during the training phase are considered as cold word. We declare a cold word as hot if its frequency is more than a threshold. Although the popular word entries in the PWT are identified during the training phase, some newly popular words continue to enter the PWT (as cold words) beyond the training phase. In such circumstances, we need to make room in the PWT by evicting the LRU entry among the existing cold entries. As this entry has been used during the PEB construction, it would create problems during the decryption of the PEBs that contain the index of this to be evicted PWT entry. We remove this probable anomaly by maintaining addresses of the memory blocks that contain the indices of cold PWT word entries. A list of addresses of the PCM blocks sharing a cold word is maintained for its corresponding PWT entry using the nodes of the pointer pool. In particular, the lists of addresses are maintained only for the cold word entries since the cold words are more prone to eviction from the PWT than the hot word entries. Also, the associated addresses that share a cold word are very few. After the end of the training phase, the PEB construction (B.1 in Figure 5.5) or PEB decryption (B.2 in Figure 5.5) takes place depending on the type of request (write/read) coming to the memory controller.

5.4.2.3 Partially Encrypted Block (PEB) Construction and Decryption

(1) PEB Construction

Algorithm 4 describes the detailed procedure of the steps to be taken once an evicted cache block (b) from LLC reaches the memory controller. The various operations related to PEB construction are discussed below.

The OTP is generated using the counter associated with the cache block (b), its address, and the key (line 8, Algorithm 4). Alongside, the words in b are searched in the PWT to identify the popular words. On finding a match in the PWT, the encryption of the

Algorithm 4: Pop-Crypt : PEB Construction

```
1  n : Total words in a cache line
2  b: Unencrypted cache line data, b_PEB : Partially encrypted cache block
3  w_i : ith word inside b,
4  w_iPEB : ith word inside a PEB,
5  addr_b : Physical address of the cache block b
6  ctr_b : Counter associated with block b for counter mode encryption
7  for Every block b coming to PCM memory do
8      OTP = AES_encrypt(ctr_b, addr_b, key)
9      for ∀ w_i ϵ b, i = 1, 2..., n do
10         if w_i hits in index j in PWT then
11             w_iPEB = j
12             BPV[i] = 1
13             UpdatePWT(j, addr_b)
14         else
15             w_iPEB = w_i XOR OTP(i)
16             BPV[i] = 0
17             UpdatePWTT(w_i)
18     Add w_iPEB as the next word in b_PEB
19 Function UpdatePWT(j, addr)
20     PWTentry[j].freq++
21     PWTentry[j].TS = curTick()
22     if PWTentry[j].HotBit == 0 then
23         if PWTentry[j].freq > HotThreshold then
24             PWTentry[j].HotBit = 1
25             Free pointer list associated with the entry j
26         else
27             Add addr to the pointer list
28 Function UpdatePWTT(w)
29     if w hits jth entry in PWTT then
30         PWTTentry[j].freq++
31         if PWTTentry.freq>Th then
32             Insert w by evicting cold LRU PWT entry after Re-encrypting PEBs and freeing the
                 pointer list associated with the LRU entry
33     else
34         Place w by replacing the least frequent entry from PWTT
```

popular word is skipped by replacing the word with its index from the PWT (line 11). A Bit Presence Vector (BPV) is created to identify the encrypted vs. non-encrypted words in the cache block. The BPV bits for the popular/non-popular words are set/reset to 1/0 (lines 12, 16). We show a pictorial example of PEB construction/decryption later in this Section for better understanding of the reader. During execution, the PWT and PWTT entries are also updated (lines 13 and 17, respectively) to keep track of the popular words. These steps are described in the next subsection.

(2) PEB Decryption
During decryption, the BPV associated with a PEB is scanned bit-wise. If BPV for a word position is 1, it indicates that the word is a popular word. Therefore, the actual word is fetched from the PWT by using the index present in the word position. On the other hand, BPV=0 indicates that the word is non-popular, and hence it was encrypted during PEB construction. The corresponding un-encrypted word is generated by XORing the encrypted word with the corresponding bits of the OTP. Finally, the decrypted block is sent to the requester LLC.

5.4.2.4 Update PWT and PWTT

During PEB construction, we continue to update the metadata in the PWT and PWTT to gather any recently becoming popular words and also update the pointer list for new addresses using the cold words. This is done for bookkeeping and to take care of the dynamic nature of application profiles.

Update PWT: On getting a hit in the PWT for a popular word w_i, the corresponding entry at index= j is updated by incrementing its $frequency$ field (line 20) and updating its *timestamp* field with the current timestamp (line 21).

- **Hit in a Cold Entry**: If the word is cold, then the address of the PCM block containing this word is added to the corresponding pointer list (line 27). If the frequency of the word exceeds a certain threshold ($HotThreshold$[1]), the word is declared hot by setting its $HotBit$ field to 1 (line 24). Our experiments reveal that the words that become hot are likely to remain hot, and therefore, they are rarely evicted from the PWT. As a result, the addresses present in the pointer list of the word entry that has newly become hot are freed (line 25) for judicious utilization of the limited size of the pointer pool. However, before the release of the address pointers, the indices of that PWT entry present in the PEBs at the corresponding memory addresses are replaced with the encrypted version of the word. Note that we do these operations in the background to lessen their impact on the system performance.

[1]We did extensive empirical analysis to set the value of $HotThreshold$=100.

- **Hit in the Hot entry**: For the hot words, the addresses of the blocks using the word need not be added to the list of address pointers as their chances of eviction from the PWT are very less. However, in the rare events of eviction of hot words from the PWT, the whole memory is encrypted using a different key of the AES. This is a routine procedure adopted in practice at the time of counter overflow in CME [113].

  System execution passes through different phases [143, 144], where each phase is marked by some repetitive program behavior in terms of performance characteristics (like IPC improvement, cache miss ratio etc) and resource requirements (Cache size, write buffer etc). Generally, the execution phases are stable and long enough with short transition periods ([143]) between the phases. Pop-Crypt adapts nicely to the change in the execution phase. It dynamically updates the PWT using the PWTT to contain the popular words belonging to the subsequent execution phase. In this process, Pop-Crypt enters new popular words by primarily evicting the cold words (followed by re-encryption of the blocks associated with the evicted entry) from the PWT. Since the PWT eviction is very low for all the benchmarks (kindly refer to Table 5.3), therefore, the eviction of the cold words alone is enough to accommodate new words in the PWT in most of the cases. However, on the rare occasion of hot word eviction, the whole memory is re-encrypted using a different key of AES, as mentioned above.

Update PWTT: On a miss in the PWT, the word w_i is encrypted (line 15) and its BPV bit is reset to 0 (line 16). As we continue to identify the newly popular words during execution, we update PWTT (line 17). On getting a hit for the word w_i in PWTT (say, in the jth entry), the $frequency$ field of the entry in the PWTT is incremented (line 30). If the frequency of the word exceeds a threshold, we consider bringing this word to PWT, assuming it to be a good candidate for placing in the PWT (line 31). The entry for the word w_i is then inserted in the PWT by evicting the cold LRU entry (line 32). In order to make judicious use of the limited pointer pool, the address pointers associated with the LRU entry must be freed. However, before doing so, the PEBs that contain the PWT index of this evicted entry must be replaced with the encrypted version of the evicted word. On getting a miss for the word w_i in the PWTT, it is placed in the free entry after removing the least frequent entry from the PWTT (line 34).

5.4.2.5 Working Example

Figure 5.6 shows a working example of PEB construction/decryption by Pop-Crypt. We take a cache block having 8 words (A, C, B, D, F, H, B, L) for illustration purposes. The initial BPV for the block is <00000000>. Out of these 8 words, words A and B are popular at indices 4 and 5 in PWT, respectively. The other fields associated with a PWT entry are not shown for simplicity. During PEB construction, the OTP for encryption is generated by the AES controller using the counter associated with the cache block, its address, and

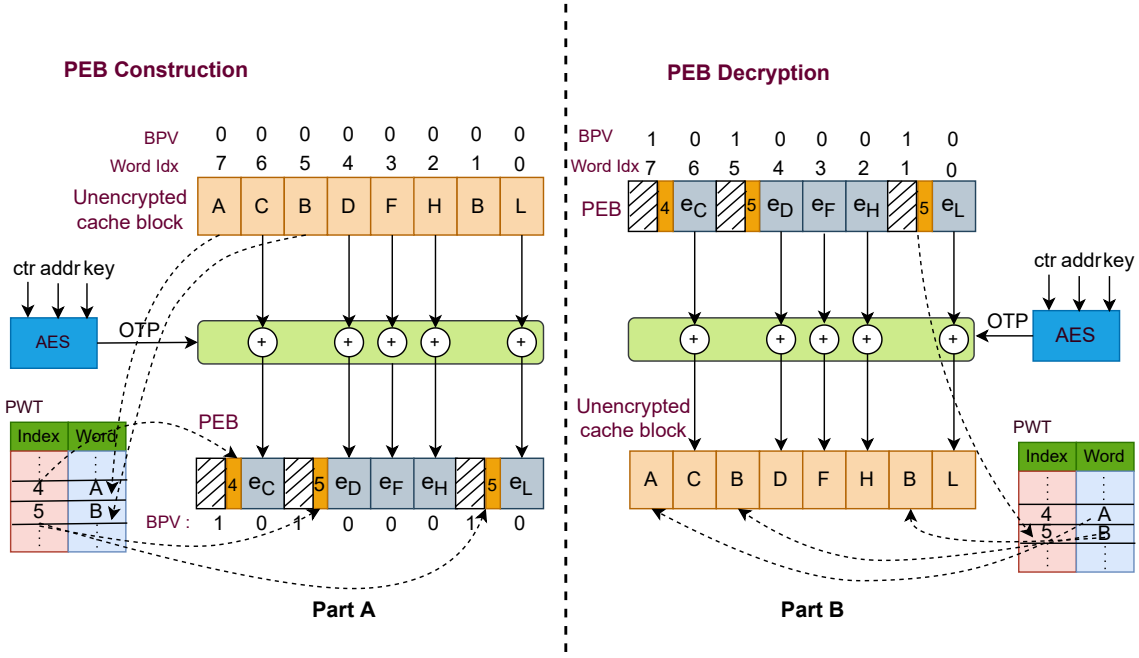


Figure 5.6: *Working Example of PEB Construction and Decryption*

the key. The non-popular words C, D, F, H and L are XORed with the corresponding bits of this OTP to get encrypted words e_C, e_D, e_F, e_H and e_L . On the other hand, the popular words A and B occurring at index positions 7 (A), 5 (B), and 1 (B) of the cache block are replaced with their PWT indices 4, 5, and 5, respectively. The generated PEB and the corresponding BPV (<10100010>) for the block are shown in part A of Figure 5.6. Note that instead of writing the full word, we only write the index using few bits (6 bits for 64-entry PWT), and the remaining portion is kept unused (shown in shaded part). The unused bits are unaltered and do not contribute to any bit-flips.

During PEB decryption, BPV associated with the block is scanned bitwise to identify the indices of the popular words. The BPV corresponding to words at indices 0, 2, 3, 4, and 6 are zero. It indicates that the words at these index positions are non-popular (hence, they were encrypted) in the original cache block. The original words are derived by XORing the corresponding encrypted words with the respective bits in the OTP. On the other hand, the BPV bits are 1 for words with indices 1, 5, and 7. It indicates that these are popular words, and hence, contain indices of their PWT entries in the PEB. The indices (4 and 5 for words A and B, respectively) are looked up in the PWT, and the corresponding original words are inserted in the block. The original block after PEB decryption is shown in Part B of Figure 5.6.

Table 5.1: *System Parameters and Benchmarks*

Processor	
CPU	2Ghz, Quad-core, X86
L1 Cache	Private, 32 KB SRAM split I/D caches, 2-way associative, 64 B block, 1 cycle latency
L2 Cache (LLC)	Shared, SRAM, 64B block, 8-way associative 10 cycle latency, Size: 8MB
Main Memory (Using PCM)	
Capacity	PCM: 4 GB, 4 channels, Memory Controller: FRFCFS, Write buffer depth : 32
Memory Latency	Read latency: 50ns, Write latency: 150ns
Memory Energy	PCM :: Read Energy (pJ/bit): 2.47, Write Energy (pJ/bit): 14.03 (Set), 19.73 (Reset)
PWT/PWTT Latency	1.78ns/2.73ns (Per access).
PWT/PWTT Energy	0.34nJ/2.94nJ (Per access).
Encryption Parameters	
AES latency [100]	96ns Per line
AES energy [100]	5.9 nJ Per 128-bit block
Counter Cache	2MB per memory controller
Benchmarks and their classification (High/Medium/Low) based on WBPKI	
Canneal(Med), Dedup(High), Freq (Low), Fluid (Low), Stream (Low), x264 (Med) **Mix1** : <gobmk, lbm, sjeng, bzip2> (High), **Mix2 :** <milc, mcf, hmmer, bzip2>, (High) **Mix3 :** <calculix, sjeng, h264ref, leslie3d> (Low)	

5.5 Experimental Evaluation

We implement our technique Pop-Crypt on a full system simulator GEM 5[124] integrated with NVMain[125], a cycle-accurate main memory simulator designed for NVMs. The system parameters used in the experiments are shown in Table 5.1. We evaluate our results using multi-threaded PARSEC[116] and multi-programmed SPEC 2006 [133] benchmark suite. Note that the SPEC mixes are composed by considering the Write-Backs per Kilo Instruction (WBKI) of each individual benchmark. We run the SPEC 2006 workloads for 1B instructions after warming them up by at least 250M instructions. We have categorized the benchmarks as High, Medium and Low write intense benchmarks, based on their Write Back Per Kilo Instruction (WBPKI) (shown in Table 5.1) We have taken the AES encryption latency and energy to be 96ns per line and 5.9nJ per 128-bit block based on the specifications [145]. The PWT/PWTT access latency and Energy are calculated using CACTI [130] tool and are reported in the Table 5.1. These latency and energy are negligible compared to read/write latency and energy (per block). So, they have little impact on overall energy consumption and system performance. We have taken into account these values during our evaluations.

We evaluate our technique Pop-Crypt with state-of-the-art techniques DCW [42], FNW [43], SECRET [40] and DEUCE [39]. We take DCW to be the baseline for our evaluations. We have also shown the effectiveness of combining FNW with Pop-Crypt. Note that all the techniques use DCW [42] to reduce redundant bit-flips.

- **Baseline (DCW)** [**42**]: Only the modified bits of the cache blocks are written to PCM to eliminate redundant bit-flips.
- **FNW** [**43**]: It inverts the data if more than half of the bits in the cache blocks are modified and bounds the maximum bit-flips to half of the number of bits in the blocks.
- **SECRET** [**40**]: It avoids the encryption of the clean and zero words with the help of local counters and bits to identify zero words (associated per word). In the event of the local counter overflow, all the local counters are reset and the entire block is re-encrypted.
- **DEUCE** [**39**]: DEUCE partially encrypts the words inside an incoming cache block that gets modified during an epoch interval while avoiding the encryption of the unmodified words.
- **Pop-Crypt**: It is our proposed technique that skips the encryption of the popular words inside the incoming cache lines by replacing the popular words with their PWT indices.
- **Pop-Crypt+FNW**: Pop-Crypt is combined with FNW to make it more effective.

5.5.1 Results and Analysis

The proposed technique Pop-Crypt, along with the baseline DCW [42], and existing techniques (FNW [43], DEUCE [39] and SECRET [40]) are evaluated in terms of bit-flips, energy, lifetime, Average Memory Access Time (AMAT) and Instructions per cycle (IPC). All the results are normalized over DCW.

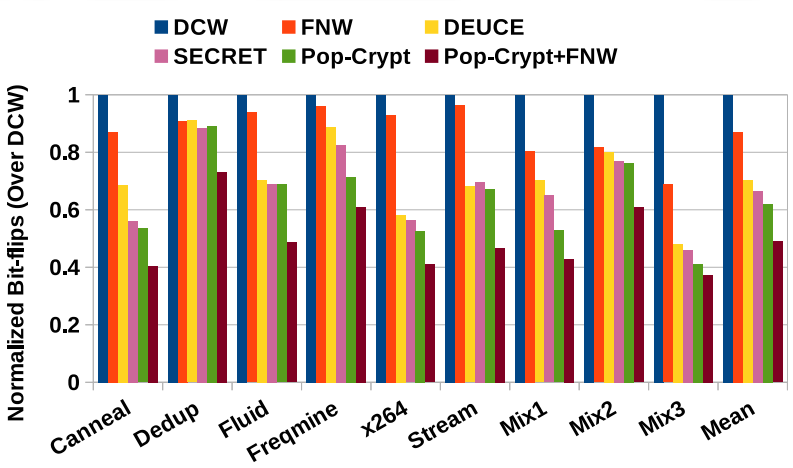


Figure 5.7: *Normalized Bit-flips over DCW (Less is better)*

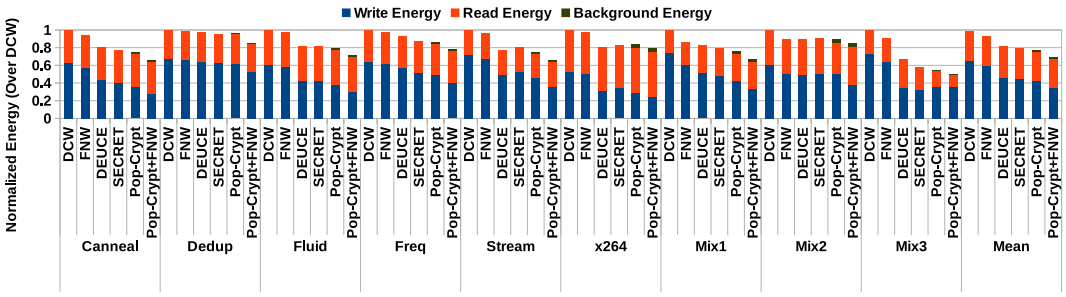


Figure 5.8: *Normalized Energy over DCW (Less is better)*

5.5.1.1 Effect on Bit-flips and Energy Consumption

Pop-Crypt avoids the encryption of popular words inside the incoming cache lines while encrypting only the non-popular words. Since the number of popular words is high, the encryption of many words is avoided. It reduces the bit-flips in the PCM cells to a great extent and mitigates the harmful implications of the Avalanche Effect of encryption. Figure 5.7 shows the normalized bit-flips of DCW, FNW, DEUCE, SECRET and our proposal Pop-Crypt along with Pop-Crypt+FNW. On average, the reduction in bit-flips (over DCW) shown by FNW, DEUCE, SECRET Pop-Crypt, and Pop-Crypt+FNW are 13%, 29%, 34%, 38% and 51%, respectively.

Pop-Crypt outperforms the existing techniques DEUCE and SECRET in reducing bit-flips. DEUCE encrypts only the words inside the incoming cache lines that get modified over an epoch of 32 writes, avoiding the encryption of the unmodified words. However, DEUCE needs occasional re-encryption of all the words at the end of the epoch interval, which increases the bit-flips. SECRET employs per word local counters and zero bits (to identify the zero words) to avoid the encryption of the clean and zero words inside the blocks. However, when a local counter corresponding to a word expires, the local counters of all the words are reset, and the entire block is encrypted using a new counter, which increases bit-flips. Also, SECRET can not perform optimally in the scenarios where the incoming cache blocks contain a high percentage of modified words. In such cases, since the encryption of the modified words can not be avoided, it leads to frequent expiry of the local counters. On the other hand, Pop-Crypt adopts a different approach by avoiding encryption of the popular words. The efficacy of Pop-Crypt lies in the availability of a large percentage of Popular words (as mentioned in Section 5.3) across the incoming cache lines. The combination of Pop-Crypt with FNW reduces bit-flips further. For some benchmarks like x264 and Mix3, the reduction in bit-flips is more as Pop-Crypt achieves a high hit rate in the PWT due to their high frequency of the hot words (with lesser hot word counts) (kindly refer to Figures 5.12, 5.2 and 5.3, respectively).

In PCM-based memories, the write energy dominates over the read energy and holds

a larger share in the overall energy consumption. Therefore, the techniques that reduce bit-flips also reduce energy consumption. Pop-Crypt reduces write energy (more compared to DEUCE, SECRET and FNW) due to reduction in bit-flips in the PCM cells. Hence, Pop-Crypt remains successful in reducing the overall energy consumption. Figure 5.8 shows the normalized energy consumption in PCM (Energy as the sum of read energy and write energy) [1]. The values of read energy and write energy per bit are shown in Table 5.1. On average, the reduction in energy consumption by FNW, DEUCE, SECRET, Pop-Crypt, and Pop-Crypt+FNW are 6%, 18%, 20%, 22% and 27%, over DCW, respectively.

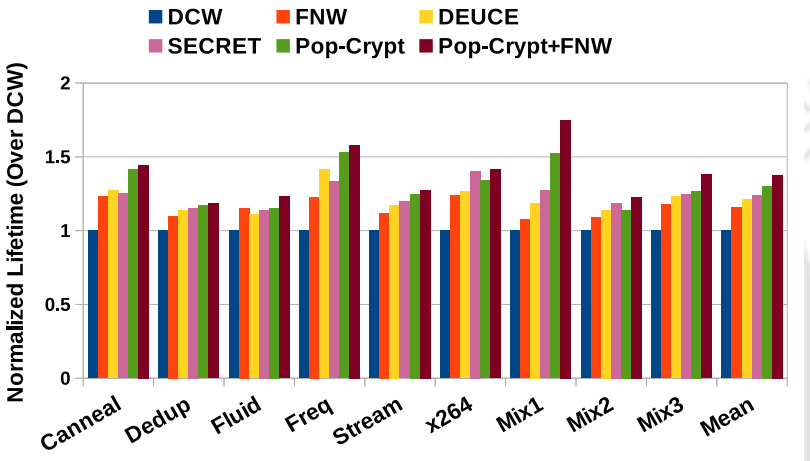


Figure 5.9: *Normalized Lifetime Over DCW (More is better)*

5.5.1.2 Effect on PCM Lifetime

PCM-based main memories have limited write endurance, i.e., they can withstand only a limited number of writes before wearing out completely. Similar to Non-volatile caches, the lifetime of PCM can be defined in two different ways: **Raw Lifetime** and **Error-Tolerant Lifetime**. In this work, we focus on improving the raw lifetime since it is the base for error-tolerant lifetime [74] (For details, kindly refer to Section 2.3.1).

Reduction in bit-flips by Pop-Crypt results in the decrease in write activities in the PCM cells. It helps in promoting PCM lifetime by relieving the excessive write pressure in the PCM cells induced by encryption. Since the reduction in bit-flips shown by Pop-Crypt is more compared to FNW, DEUCE and SECRET; therefore Pop-Crypt gets a clear edge in

[1]Note that we have shown the background energy consumption for Pop-Crypt. Background energy is the energy consumed during the eviction of cold entries from PWT, which leads to writing the re-encrypted versions of the PEBs that share the evicted PWT entry in the PCM. Background energy is negligible compared to read and write energy due to lesser PWT evictions from the PWT, as confirmed by the experiments, shown in Table 5.3.

improving PCM lifetime to a greater extent compared to FNW, DEUCE and SECRET. The addition of FNW over Pop-Crypt enhances PCM lifetime further for most benchmarks. However, for dedup, the lifetime improves marginally for Pop-Crypt+FNW over Pop-Crypt since the variation of bit-flips in the words inside the cache lines of dedup is less uniform than the other benchmarks. On average, the percentage improvement in lifetime shown by FNW, DEUCE, SECRET, Pop-Crypt, and Pop-Crypt+FNW are 15%, 21%, 24%, 29%, and 36%, over DCW, respectively (Shown in Figure 5.9).

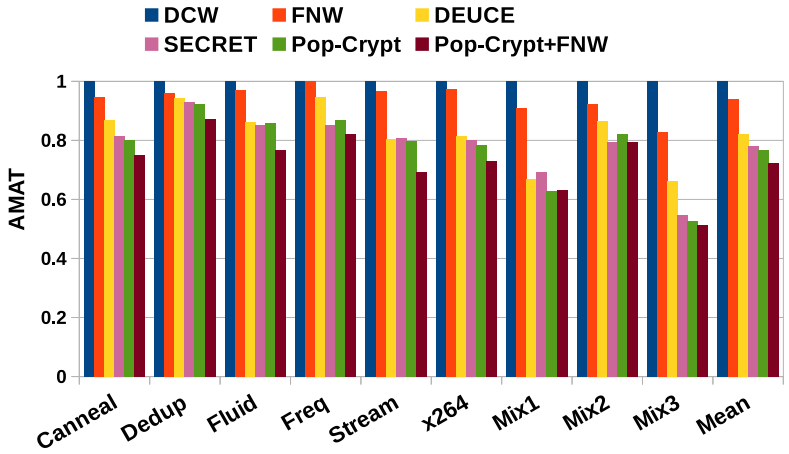


Figure 5.10: *Normalized AMAT over DCW (Less is better)*

5.5.1.3 Effect on Average Memory Access Time (AMAT)

An incoming cache block is written in PCM arrays over multiple write slots ([39]) since the current available for writing is not enough to write all the bits of the block at one go. Since the reduction in bit-flips reduces the number of bits to be written in the PCM cells, therefore, it can speed up the write service latency by reducing the number of slots required to write a cache line in PCM. Reduction in write service latency also reduces read service latency by reducing the memory contention for reads. Reduction of read/write latencies reduces the Average Memory Access Time (AMAT), which helps in improving the system performance. Pop-Crypt outperforms FNW, DEUCE and SECRET due to its ability to reduce write slots while performing the writes of cache lines in PCM arrays. In particular, the reduction in average slots [1] per write by FNW, DEUCE, SECRET, Pop-Crypt, and Pop-Crypt+FNW are 10%, 15%, 22%, 27%, 30%, respectively, over DCW. On average, the reduction in AMAT shown by FNW, DEUCE, SECRET, Pop-Crypt, and Pop-Crypt+FNW are 6%, 18%, 21%, 23% and 27%, over DCW, respectively (shown in Figure 5.10).

[1]We take write-width per slot to be 128 bits; therefore, the write of a cache line of 64 bytes can take up to a maximum of 4 slots, as in [39].

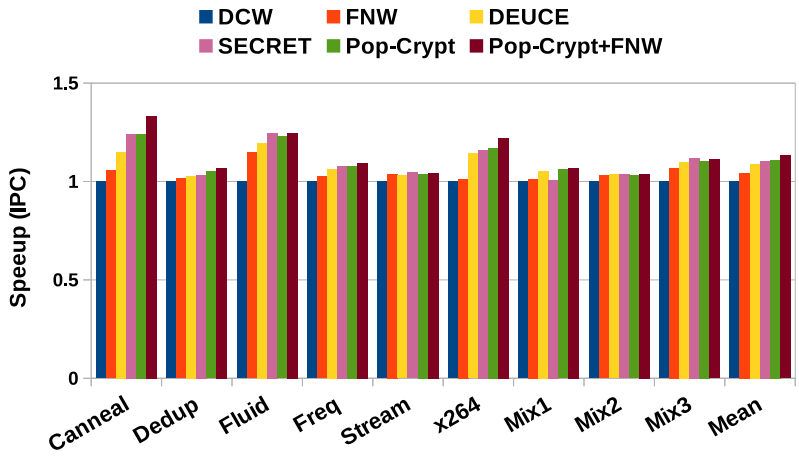


Figure 5.11: *Normalized Speedup over DCW (More is better)*

5.5.1.4 Effect on Performance

As mentioned above, reduction in bit-flips minimizes the latency involved in writing a cache block in PCM arrays. Pop-Crypt reduces bit-flips in encrypted PCM by skipping the encryption of the popular words. Since popular words are highly prevalent in all the workloads (as backed up by our experiments, discussed in Section 5.3), therefore, Pop-Crypt is able to maintain a fair hit rate in the PWT for all the workloads with varying write intensity (measured in terms of WBPKI) (kindly refer to Figure 5.12). As it is evident from Figure 5.12, Pop-Crypt is robust enough to maintain a fair hit rate in the PWT even under high WBPKI scenarios. As a result, Pop-Crypt reduces bit-flips to a high extent by skipping the encryption of the popular words, leading to reduced latency while servicing the writes. Servicing the writes quickly reduces the main memory write cycle latency. Not only that, it reduces the memory contention for reads. It helps in improving the system performance, which is measured in terms of Instruction Per Cycle (IPC). As discussed above, Pop-Crypt reduces the write service latency, resulting in the improvement of IPC. On average, the improvement in IPC shown by FNW, DEUCE, SECRET, Pop-Crypt, and Pop-Crypt+FNW are 4%, 8%, 9%, 11% and 13%, respectively (shown in Figure 5.11).

5.5.2 Sensitivity Analysis

Below, we give sensitivity analysis on the size of PWT, interval length of training phase, word-size and capacity of the pointer pool.

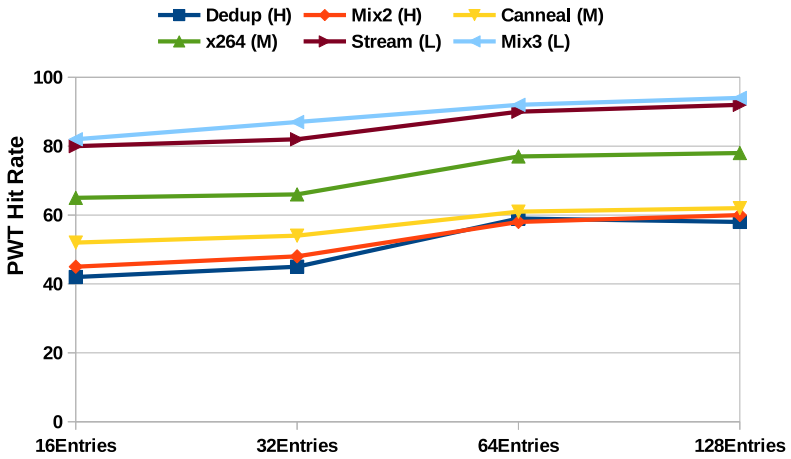


Figure 5.12: *Variation of PWT Hit Rate with the Size of PWT (H, M, L indicates High, Medium, Low Write Intense Benchmarks, respectively)*

5.5.2.1 Size of PWT

The popular words can be accommodated in a small-sized PWT since their count is very few (illustrated in Section 5.3) across the incoming blocks. Increasing the size of PWT beyond a certain size brings no significant benefits since the few popular words that are stored in the additional space remain as dead words and are accessed rarely. Figure 5.12 shows the variation of PWT hit rate for PWT size 16, 32, 64, and 128 entries for some representative benchmarks. We have selected two benchmarks, each from the high, medium, and low write intense category, as shown in Table 5.1. The hit rates are higher for the benchmarks with low/med WBPKI (like Stream, Mix3, x264) and vice-versa. However, all the benchmarks show fair hit rates, justifying the robustness of Pop-Crypt for different scenarios of write intensity. A higher PWT hit rate indicates that the words stored in PWT are accessed with high frequency. It is evident from the figure that the hit rate initially increases as the size of the PWT increases and becomes stable after 64 entries. This size is sufficient even for the benchmarks with high WBPKI since the number of popular words is still less for these benchmarks, as evident from our experiments. We found experimentally that the hit rate obtained using a 256-entry PWT for dedup and Mix2 (high WBPKI) is around 70% and 75%, respectively. However, keeping in view the high storage overhead in the memory controller for such large PWT, we decided to keep the size of PWT as 64 entries for these benchmarks, as well. It shows that 64 entries are sufficient to hold the vital popular words in the PWT. Hence, we have decided to take PWT size of 64 entries for our experiments.

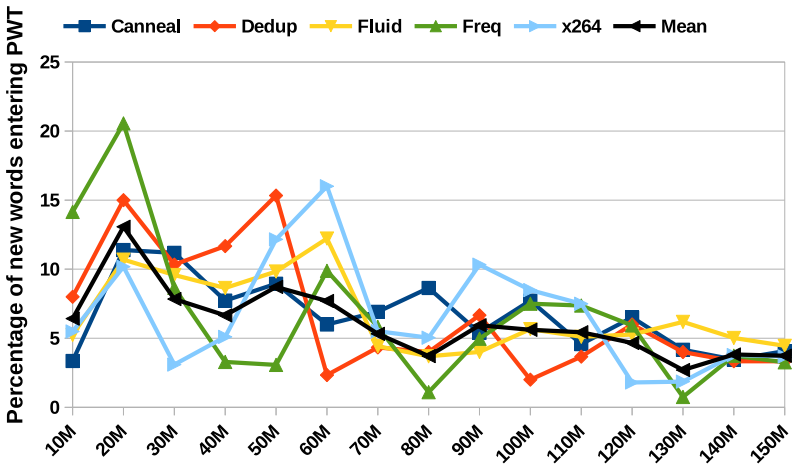


Figure 5.13: *% of New Words Inserted over Time Interval*

5.5.2.2 Training Phase Interval Length

We found experimentally that a large number of newly popular words insert initially after the application execution starts, and this number decreases slowly as the execution continues. Figure 5.13 shows the percentage of new words entering PWT up to 150M cycles (from the beginning of application execution) at different time intervals. It is evident that the percentage of new words entered is very high initially but slowly decreases later. This is because, as we stated in section 5.3, the words that are repeated frequently across the cache lines are very few, and they enter PWT in the initial periods of the application execution itself. For most of the benchmarks, the entry of new words in PWT shows a declining trend after around 100M cycles after the start of the application execution.

Table 5.2 shows the percentage ratio of the training phase length over the total length of application execution. Note that this percentage ratio varies across the benchmarks since the total execution cycles vary across the benchmarks (For example, x264 and canneal are the shortest and longest benchmark, respectively). We can see that the training phase occupies only a small fraction of the overall application execution.

5.5.2.3 Word Size

Pop-Crypt reduces more bit-flips for popular words of larger sizes. This is because skipping encryption of large-sized words reduces more bit-flips compared to small-sized words. Also, overhead due to the BPV bits is lesser for larger words since the lesser BPV bits are required for blocks having larger words. However, the PWT and PWTT entries, which are maintained in the memory controller, occupy more space for large-sized words (%reduction in bit-flips, BPV overhead, and PWT+PWTT overhead are shown in Tables 5.4 and 5.5, respectively

Table 5.2: *Percentage ratio of training phase length (TL) to the application execution length (AL)*

Benchmarks	% Ratio of TL/AL
Canneal	0.62
Dedup	3.5
Fluid	4
Freq	1.43
Stream	4.3
x264	7.5
Mix1	7.4
Mix2	3.84
Mix3	7.24
Mean	**3.51**

Table 5.3: *Percentage PWT evictions per 100 writes*

Benchmarks	%PWT evictions
Canneal	0.02
Dedup	0.13
Fluid	0.98
Freqmine	0.58
Stream	0.35
x264	1.29
Mix1	1.69
Mix2	3.65
Mix3	0.27
Mean	**0.47**

for word size 16, 32, 64 bits). In order to keep low storage overhead (due to PWT+PWTT) in the memory controller yet maintaining a fair reduction in bit-flips and low BPV storage overhead, we have taken the word size of 32 bits for our experiments.

5.5.2.4 Capacity of the Pointer Pool

The addresses of the memory blocks sharing the cold words in PWT are maintained as lists for each cold word entry. These lists are formed using the nodes present in the pointer pool. However, the pointer pool should be limited in size in order to reduce the storage overhead in the memory controller. Our experiments reveal that the percentage of PWT evictions due to the insertion of newly popular words in the PWT is negligible after the training period (as shown in Table 5.3). For Mix benchmarks, the popular words belonging to the constituent workloads get collected during the training period. Since the constituent benchmarks have only a few popular words, the space in the PWT can sufficiently hold the popular words belonging to these Mix benchmarks, leading to lesser PWT eviction. Therefore, we decide to keep the addresses of only 1000 words in the pointer pool shared by the cold entries in the PWT.

5.5.3 Overhead Analysis

Pop-Crypt uses 28-bit line counters for CME similar to [39]. The size of the BPV is 16 bits per cache line. Therefore, the storage overhead [1] due to the BPV bits is only 3.12%. The

[1]Compared to the baseline that applies DCW on encrypted memory using CME with 28-bit counters per line

Table 5.4: *Percentage reduction in Bit-flips by Pop-Crypt over DCW for different word sizes*

	16-bit	32-bit	64-bit
Canneal	44%	47%	51%
Dedup	11%	19%	57%
Fluid	31%	35%	59%
Freq	29%	36%	57%
Stream	32%	43%	60%
x264	43%	48%	64%
Mix1	36%	49%	52%
Mix2	17%	24%	28%
Mix3	56%	58%	62%
Mean	**30%**	**38%**	**53%**

Table 5.5: *Storage overhead due to PWT, PWTT and BPV bits for different word sizes*

	16-bit word	32-bit word	64-bit word
for PWT +PWTT	856+6208 =7064B	984+7232 =8216B	1240+9280 =10520B
For BPV	6.25%	3.1%	1.56%

storage overheads of DEUCE (6.25% with word size 16 bits) and SECRET (6.25% with word size 64 bits) are higher compared to Pop-Crypt since DEUCE uses fine granularity words (2B word size), whereas SECRET use per word local counters. Also, unlike DEUCE[39] and SECRET[40] that use multiple AES engines, Pop-Crypt uses only one AES engine. Based on the highly optimized energy-efficient hardware implementation given by Mathew et al. [146], the area overhead due to AES is $0.02mm^2$ at 22 nm technology node. Following standard Intel i7 die size ($160mm^2$) at 22nm [147], the area overhead due to the AES engine is 0.0125%, which is negligible in practice.

Storage Overhead in the Memory Controller: As shown in Figure 5.4, a PWT entry consisting of ⟨`Word(32),Frequency(16),Timestamp(64), HotBit(1), Pointer to address list(10)`⟩ has a length of 123 bits. Therefore, the total space required for 64 entry PWT is 984 bytes. On the other hand, a PWTT entry has a length of 113 bits. Note that the fields in the PWTT field are the same as the PWT, except the 10-bit pointer field is not present in the PWTT entry. Therefore, a 512-entry PWTT consumes a size of 7232 bytes. Finally, the pointer pool that stores a total of 1000 addresses (each address is of size 64 bits) of the cold words needs a total capacity of 8000 bytes. The total storage required in the memory controller is $984 + 7232 + 8000 = 16216$ bytes$\approx 16KB$.

5.6 Summary

Non-volatile memories are promising candidates for constructing high-density and energy-efficient main memories. However, the non-volatility feature of the NVMs opens the door to data confidentiality-based attacks like Bus-snooping and stolen DIMM attacks. Encryption techniques that are used to provide safety to the NVM data lead to enormous bit-flips in the NVM cells, which degrades the NVM lifetime severely.

In this chapter, we propose a technique called Pop-Crypt that avoids the encryption of the popular words, which are repeated with high frequencies across several cache blocks, while encrypting only the non-popular words. The generated partially encrypted blocks reduce bit-flips, energy consumption and improve PCM lifetime. It also improves system performance due to reduction in write service latency. On average, Pop-Crypt reduces bit-flips by 38%, 29%, 11%, 6%; reduces energy consumption by 22%, 16%, 5%, 4% and improves lifetime by 29%, 13%, 7%, 5% over DCW, FNW, DEUCE and SECRET respectively. Thus, seeking newer avenues related to the identification and management of data similarity can assist in longevity enhancement of the encrypted PCM-based main memories.

6

Exploring Newer Avenues of Bit-flip Reduction in Encrypted NVM Using Compression and Encoding

In the last chapter, we presented Pop-Crypt, a partial encryption-based method that bypasses encryption of the words that frequently repeat inside blocks to minimize bit-flips in encrypted PCMs. This chapter explores additional ways based on compression and encoding for minimizing bit-flipping in PCMs while preserving memory security via encryption. In particular, we offer two approaches that compress the data blocks prior to encryption to mitigate the avalanche effect (refer to Section 2.2.2) of encryption. The first method, called CoSeP utilizes compression techniques FPC, BDI, and COMF (COMF is our technique proposed in Chapter 3) to get an optimum balance of compression ratio and coverage. On the contrary, CADEN, the second technique, encodes the compressed and encrypted blocks before writing to reduce bit-flips.

6.1 Introduction

Encryption provides strong security guarantee to the sensitive data stored in NVMs against data confidentiality based attacks (kindly refer to Section 2.2.2). However, most of the existing encryption techniques show diffusion property [39, 101] i.e., changing even a single bit in the plain text leads to enormous bit-change in the cipher text [39], which severely

degrades the NVM lifetime. It renders the existing encoding techniques like DCW [42], FNW [43] obsolete in the presence of encryption, particularly due to the *coarse* encoding granularity (Please refer to Section 4.3.2.1 for the definition of encoding granularity) adopted by these techniques. In contrast, encoding using *fine* granularity could bring more reduction in bit-flips. But, it is not affordable as high meta-data overhead is incurred to achieve fine granularity. Compression-based approaches [51, 101, 102] also need improvisation as the existing compression techniques like BDI and FPC can not optimize *Compression Ratio* (CR) and *Coverage* simultaneously (Refer to Section 2.3.3 for definition). For instance, BDI offers a good CR but a low coverage, whereas FPC offers a large coverage but a poor CR. Our proposed technique COMF balances this trade-off to a large extent, as discussed in chapter 3. However, a judicious blending of these compression techniques could offer even further balance in CR and coverage.

In this chapter, we present methods that demonstrate potential new avenues for controlling the bit-flip surge brought on by encryption. Specifically, our proposed solutions aim to mitigate the trade-offs between CR/coverage of compression and fine granularity encoding/storage overhead of the encoding techniques. The first technique, called CoSeP, intelligently integrates the existing compression techniques FPC and BDI with our proposed compression technique COMF to obtain an optimum balance between CR and coverage. In contrast, the second technique, called CADEN, encodes the compressed and encrypted blocks at finer encoding granularity, incurring lower storage overhead to reduce bit-flips.

6.2 Chapter Overview

6.2.1 Contributions of the Chapter

The main contributions of the chapter are as follows :

- We propose a technique called CoSeP that makes a greedy choice by selecting the smallest block generated by BDI, FPC, and COMF to get optimum balance between CR and coverage.
- We propose a technique called CADEN that adaptively encodes the compressed+encrypted blocks by assigning the tag bits dedicated for the entire block to the compressed data bits only, resulting in finer granularity.

6.2.2 Chapter Organization

The rest of the chapter is organized as follows. CoSeP and CADEN are described in detail in Sections 6.3 and Section 6.5, respectively. The specifics related to CoSeP like Observations, Working principle and Experimental evaluation are presented in Sections 6.3.1, 6.3.2 and

6.4, respectively. On the other hand, Proposed methodology and experimental evaluation of CADEN are described in Sections 6.5.1 and 6.6, respectively. Finally, summary of the chapter is presented in Section 6.7.

6.3 Proposed Method 1 - CoSeP : Compression and Content-based Selection Procedure to Improve Lifetime of Encrypted NVM

CoSeP considers two important parameters, *1) Compressed Block Size (CBS)* and the *2) Content* of the compressed blocks to reduce bit-flips in the encrypted PCM. In particular, CoSeP is based on the efficient use of three state-of-the-art compression techniques : FPC [89], BDI [90] and COMF [148, 149], which vary in terms of CR and coverage. CoSeP selects the optimal technique depending on CBS of the blocks produced by FPC, BDI, and COMF. If the CBSs of the two smallest blocks differ by a large margin, then the smallest block can result in minimum bit-flips upon writing in PCM as its size is considerably smaller. In such cases, CoSeP makes a greedy choice by selecting the smallest block. In contrast, if the two CBSs are similar, it is beneficial to compute bit-flips of the compressed data with the old content. Since the resultant compressed blocks' data contents are different, these similar-sized blocks lead to different bit-flips on writing after encryption. CoSeP utilizes such opportunities by selecting the technique that leads to lesser bit-flips.

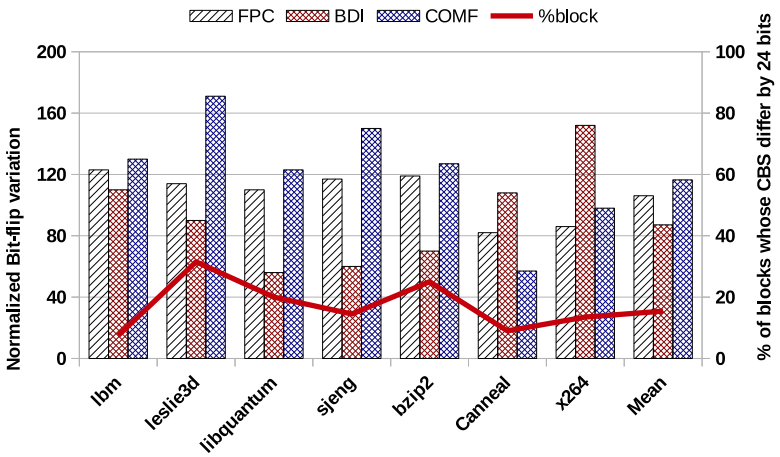


Figure 6.1: *Bit-flip Variation and Percentage Count of Blocks whose CBS differ by Δ=3 bytes*

Table 6.1: *Compression Ratio and Coverage of FPC, BDI, COMF and Selective*

Benchmark		FPC	BDI	COMF	Selective
lbm	CR	0.7	0.56	0.59	0.4
	Cov	89	20	59	99
leslie3d	CR	0.3	0.29	0.28	0.18
	Cov	70	64	82	93
libquantum	CR	0.35	0.27	0.22	0.16
	Cov	95	90	85	95
sjeng	CR	0.3	0.28	0.11	0.06
	Cov	92	60	80	99
bzip2	CR	0.48	0.35	0.25	0.22
	Cov	72	22	58	94
Canneal	CR	0.57	0.42	0.54	0.5
	Cov	85	36	82	95
x264	CR	0.32	0.25	0.22	0.15
	Cov	90	67	85	97
Mean	**CR**	**0.41**	**0.33**	**0.27**	**0.19**
	Cov	**84**	**45**	**75**	**95**

6.3.1 Observations

CoSeP is based on two observations, which are as follows.

Observation 1 : *An intelligent selection among the state-of-the-art techniques FPC, BDI, and COMF helps in achieving better Compression Ratio (CR) and Coverage.* Compression techniques minimize bit-flips by reducing the size of the incoming blocks to PCM. However, the existing techniques FPC, BDI, and COMF differ in terms of CR and *Coverage* (as shown in Table 6.1). The trade-off between CR and *Coverage* can be balanced by selecting the appropriate technique at runtime, which gives the smallest block. This is evident from the column named 'Selective' in the table, which shows better CR (=0.19) and Coverage (=95%). It indicates that most of the blocks are compressed, and the resultant blocks are sufficiently small.

Observation 2 : *A considerable percentage of blocks compressed using FPC, BDI, and COMF are of comparable sizes, and they lead to different bit-flips upon encryption.* The line graph in Figure 6.1 shows the percentage of compressed blocks produced by FPC, BDI, and COMF whose sizes differ by at most Δ=3 bytes [1]. On average, 30% of the compressed blocks produced by FPC, BDI, and COMF lie in close proximity of 3 bytes. However, the data contents of the generated blocks are different (Example given in Figure 6.3). As the contents are different, it leads to varying bit-flips after encryption. Figure 6.1 also shows

[1]We selected Δ= 3 bytes based on the analysis given in Section 6.4.1.

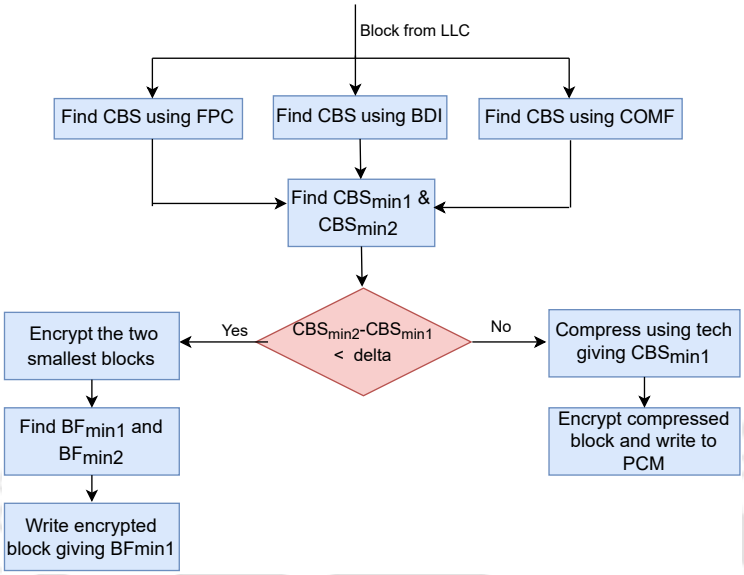


Figure 6.2: *Flowchart describing the Working of CoSeP; (CBS_{min1}/CBS_{min2} : Smallest/Second CBS, BF_{min1}/BF_{min2} : Least/Second Bit-flips)*

the normalized bit-flips[1](in bars) shown by FPC, BDI, and COMF. It demonstrates the dis-similarity in the bit-flips upon encrypting similar-sized compressed blocks.

6.3.2 Working principles of CoSeP

CoSeP optimizes CR and *Coverage* by selecting the appropriate compression technique. Flowchart in Figure 6.2 illustrates the working of CoSeP.

Encoder (Write Path): When a block evicted from the Last Level Cache (LLC) reaches the memory controller, CBS due to FPC, BDI and COMF are determined, and two smallest block are compared.

1) If the difference in size of two smallest blocks is $\geq \Delta$ *(3 bytes in our experiments)*, then we choose the block with the least CBS. We keep two compression bits (00: FPC, 01: BDI, and 10: COMF) that indicate the technique used to compress the block. Also, we keep a 9-bit CBS field per block that tells the size of the compressed blocks. These fields are required during the decryption/decompression process. The compressed block is encrypted and written in the PCM arrays.

2) If the difference in size of two smallest blocks is $< \Delta$ *(3 bytes in our experiments)*, then we assume that the blocks are of similar sizes. The blocks are encrypted parallelly using the two AES controllers, and their bit-flips with the old memory content is determined. The compressed block that leads to least bit-flips is chosen. The tag bits related to CBS are

[1]It is the bit-flips for the compressed blocks whose sizes differ by Δ=3 bytes per block

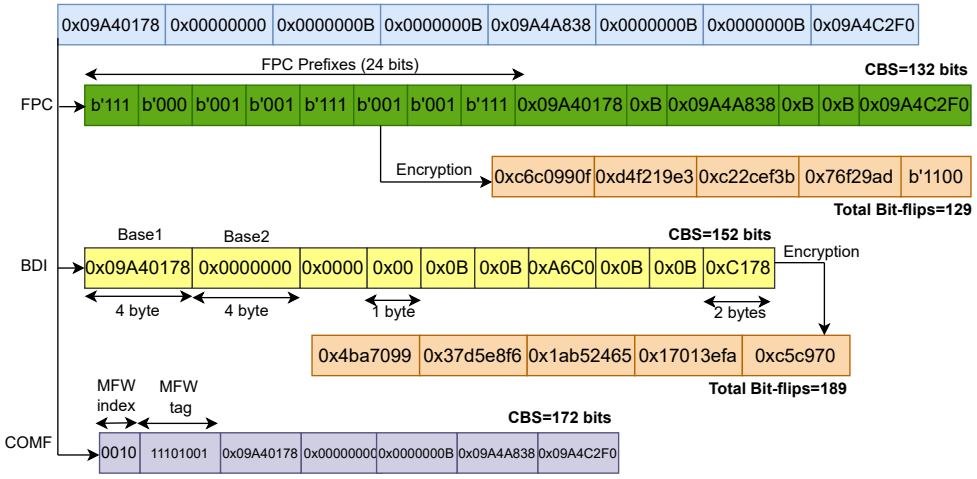


Figure 6.3: *Working Example of CoSeP*

maintained along the block, and the compression bits are stored in the compressed block in the saved space.

Decoder (Read Path): When a read request for a given block arrives at the memory controller, the compressed and encrypted block is decrypted up to the CBS. Then, based on the compression bits, the block is decompressed using the corresponding compression technique. Finally, the block is returned to the requester LLC.

6.3.2.1 Working Example of CoSeP

Figure 6.3 shows the versions of a block when compressed using FPC, BDI, and COMF. The contents of the compressed blocks are entirely different based on the compression technique used. The two smallest blocks are produced by FPC and BDI, whose sizes differ by $152 - 132 = 20$ bits ($< \Delta$). CoSeP encrypts these compressed blocks independently[1]. The bit-flips due to FPC and BDI for a particular memory block are 129 and 189 bits upon encryption, respectively. Hence, CoSeP selects the FPC compressed+Encrypted block and writes it in the PCM arrays, with compression and CBS bits as <00> and <10000100>, respectively.

6.4 Experimental Evaluation : CoSeP

Simulation Framework. We implement CoSeP on GEM5 [124] integrated with NVMain [125]. The processor frequency is 2Ghz. L1 I/D caches are two-way associative with 32KB capacity with one cycle latency. L2 cache is 8-way associative, 4MB in size with 10 cycle latency. It

[1]contents of the encrypted blocks are obtained by running AES-based CME independently on the FPC and BDI compressed blocks during Gem5+NVMain simulation

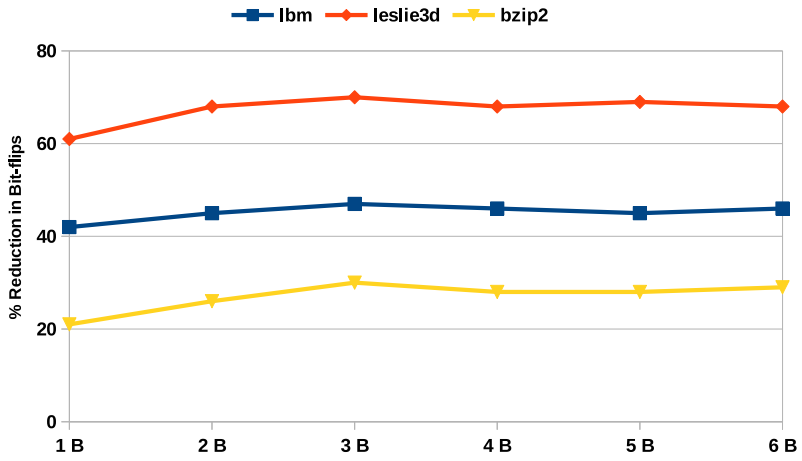


Figure 6.4: *Percentage Reduction in Bit-flips for Varying* Δ

is shared among the cores, while L1 caches are private to the cores. We model PCM with 4 channels, with total capacity of 4GB. The memory controllers use the FRFCFS scheduling policy. The latency/energy of PCM are 50ns/(2.47pJ/bit) for read and 150ns/(<14.03pJ/bit (SET),19.73pJ/bit (RESET)>) ([21, 51]) for write.

Workloads. The results are evaluated using multi-threaded PARSEC [116] and multi-programmed SPEC 2006 [133] benchmark suite. The selected benchmarks are diverse in terms of memory intensity (<MPKI, WBKI> for PARSEC and SPEC benchmarks are <2.23, 1.61> and <10.38, 7.74>). We run each SPEC workload for 1B instructions after warming them up by at least 250M instructions.

Comparison with Prior Work. We compare CoSeP with DEUCE [39], FPC [89], BDI [90] and COMF [148]. Our CompressionOnly approach chooses the block with minimum CBS generated by FPC, BDI, and COMF, whereas CoSeP further reduces bit-flips by selecting the compressed block that leads to minimum bit-flips when two minimum CBSs are comparable. The improvements are shown over DCW [42] set as the baseline for encrypted memory.

6.4.1 Sensitivity Analysis on Δ

Figure 6.4 shows the reduction in bit-flips over DCW for various values of Δ (Δ=1B, 2B,...,6B) for three representative benchmarks. Reduction in bit-flips increases initially up to Δ= 3 bytes and then becomes stable. It indicates that for higher values of Δ, the extra benefit gained by selecting blocks (based on bit-flips) becomes stable. Hence, we set Δ= 3 bytes as the cut-off value, after which CoSeP utilizes its greedy approach of selecting the most compressed block.

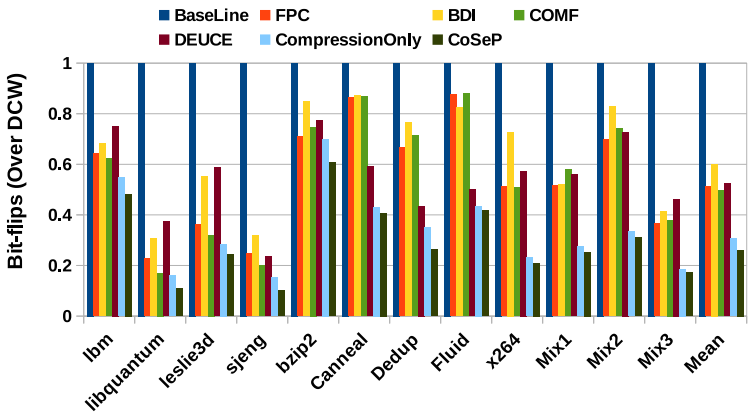


Figure 6.5: *Normalized Bit-flips over DCW (Lower is better)*

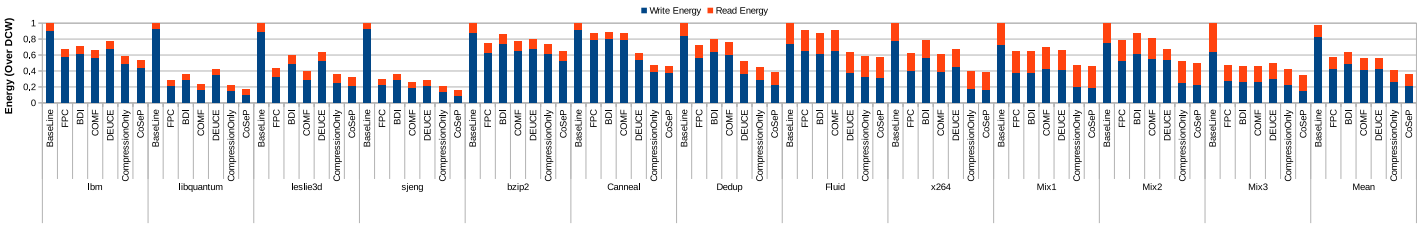


Figure 6.6: *Normalized Energy over DCW (Lower is better)*

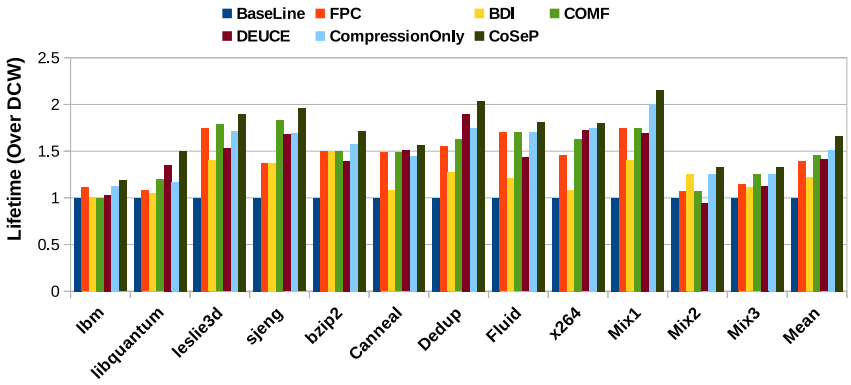


Figure 6.7: *Normalized Lifetime over DCW (More is better)*

6.4.2 Results and Analysis

1. Effect on Bit-flips and Energy Consumption : CoSeP optimizes CR and *Coverage* by integrating FPC, BDI, and COMF. It reduces bit-flips as the blocks get compressed significantly. CoSeP further reduces bit-flips by selecting the compressed block that leads to minimum bit-flips when the difference between the two smallest compressed blocks is marginal. Hence, it outperforms FPC, BDI, and COMF. It also shows more reduction in bit-flips than DEUCE since DEUCE leads to occasional re-encryption of the blocks, which increases bit-flips, whereas, CoSeP does not perform any re-encryption. Some benchmarks like libquantum, sjeng, leslie3d show more reduction in bit-flips due to their fairly low CR (As shown in Table 6.1).

CoSeP remains successful in reducing energy consumption due to its ability to reduce bit-flips to a great extent, as write energy occupies a major portion in the total PCM energy consumption. CoSeP also reduces the energy involved in encryption by encrypting only the compressed portion of the blocks. On average the reduction in <Bit-flips, Energy> by FPC, BDI, COMF, DEUCE, CompressionOnly and CoSeP are <49%, 44%>, <38%, 35%>, <52%, 46%>, <49%, 44%>, <67%, 60%> and <74%, 65%>, respectively (shown in Figures 6.5 and 6.6).

2. Effect on Lifetime : Lifetime can be defined as **Raw Lifetime** and **Error-Tolerant Lifetme**. In this work, we focus on improving the raw lifetime since it is the base for error-tolerant lifetime (Please refer to Section 2.3.1 for definition of lifetime). Reduction in bit-flips relieves write pressure in the cells, improving lifetime. Improvement in lifetime shown by FPC, BDI, COMF, DEUCE, CompressionOnly and CoSeP over DCW are 43%, 21%, 50%, 49%, 55%, and 69%, respectively (shown in Figure 6.7).

3. Cycles Per Instruction (CPI): Compression helps in faster delivery and writing of the requested blocks on read and writes, respectively. It reduces the large main memory read/write cycles and improves CPI. The optimal CR/*Coverage* provided by CoSeP and its

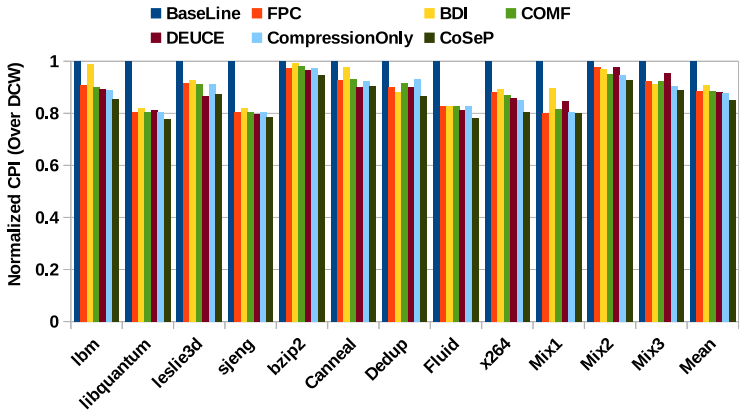


Figure 6.8: *Normalized CPI over DCW (Lower is better)*

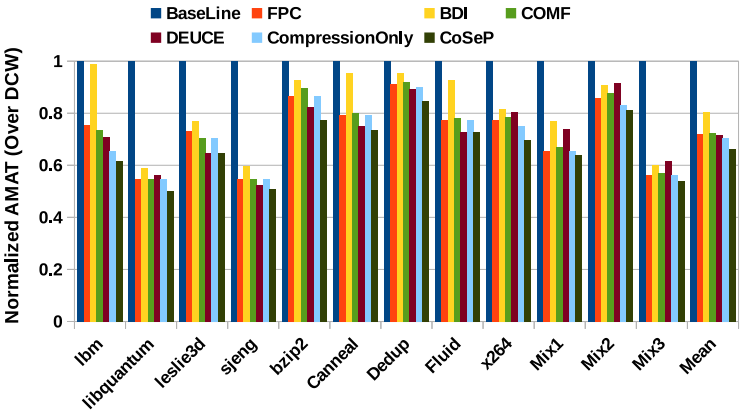


Figure 6.9: *Normalized AMAT over DCW (Lower is better)*

ability to reduce write bits improves its access speed than DEUCE, which results in better CPI. CoSeP also reduces encryption latency since encrypting compressed blocks incurs lesser latency.

4. Average Memory Access Time (AMAT): Compressed blocks can be delivered quickly from main memory, which reduces LLC miss penalty and improves AMAT. CoSeP reduces AMAT more compared to the other techniques due to its high compressibility and *Coverage*. On average, the improvement in <CPI,AMAT> shown by FPC, BDI, COMF, DEUCE, CompresionOnly and CoSeP are <12%,26%>, <10%,18%>, <14%,27%>, <13%,29%>, <12%,28%>, <16%,34%> respectively over DCW (Figures 6.8 and 6.9).

Table 6.2: *Comparison of Latency, Area and Energy of FPC, BDI and COMF*

		FPC	BDI	COMF
Latency (ns)	Compression	2	2	3.5
	Decompression	1	-	2.65
Energy (pJ/512 bits)	Compression	2.1	3.9	2.75
	Decompression	1.2	-	0.16
Area (mm2)	Total Area	0.0258	0.014	0.002

6.4.3 Overhead Analysis of CoSeP

A. Storage Overhead : CoSeP needs 9 bits to store the size of the compressed blocks, accounting for a storage overhead of 1.75%. Note that compression bits do not contribute toward overhead as they are stored in the saved space within the compressed blocks.

B. Hardware Overhead Analysis:

1. Compression/decompression : The overhead of COMF Compressor/Decompressor are determined using synthesizable Verilog code and synthesis using Genus tool from Cadence (15nm technology node). The values for FPC and BDI are taken from [51, 90]. BDI has low decompression overhead in terms of latency and energy, so these values are negligible, as per [90] (marked by the symbol "-") (Refer to Table 6.2). We discuss the overhead associated with the compression/decompression process under the effect on latency, area and energy consumption.

- Effect on Latency : Since the three compression techniques can determine CBS in parallel, the worst-case compression latency is the largest of these latencies. On the other hand, decompression can be done based on the compression bits.
- Effect on Energy Consumption : During compression, the energy consumption will be sum of all the three compression algorithms as they are run in parallel. However, during decomrpession, energy consumption will be specific to only the algorithm that was used to compress the block. CoSeP approach can be re-designed by using more efficient compression algorithms that consume less energy during compression to cut down the energy consumption cost.
- Effect on Area : The three compression/decompression engines altogether will occupy almost 0.0418 mm^2 area in the memory controller (Shown in table 6.2). However, the area overhead considering all the compression/decompression engine is still less (1.4%) for a standard memory controller having size of 2.8mm^2, as mentioned in papers [135, 136]

2 . Encryption : CoSeP uses two AES controllers to encrypt the two minimum blocks whose sizes differ marginally. We set the latency of AES encryption to 96ns per line [100]. We use

Table 6.3: *Reduction in Bit-flips for different Encoding Granularity.* $< G, T >$ *represents* $Gran_{static}$ *and number of the corresponding Tag bits.*

Benchmark	$< G, T >$		
	<64,8>	<32,16>	<16,32>
lbm	20	22	27
leslie3d	27	31	36
libquantum	25	34	46
sjeng	38	45	54
canneal	23	26	28
dedup	32	35	37
fluid	50	54	57
freq	41	46	49
x264	34	38	41
Mean	**31**	**36**	**40**

a 2MB counter cache in the memory controller to store the counters of the recently accessed blocks to avoid fetching of counters from main memory.

6.5 Proposed Method 2 : CADEN : Compression Assisted ADaptive Encoding to improve lifetime of Encrypted Non-Volatile Main Memories

CADEN utilizes compression and fine granularity encoding at low storage overhead. We use our compression algorithm COMF([148]) to reduce the effective size of the incoming cache blocks before writing in PCM. The compressed blocks are then encrypted using AES-based encryption. The compressed blocks reduce bit-flips in PCM. Subsequently, CADEN encodes the compressed and encrypted blocks at a finer encoding granularity by assigning the tag bits to the compressed data, which further reduces bit-flips.

We take FNW for our adaptive encoding approach due to its low complexity. FNW divides the data bits into partitions and assigns one tag bit to each partition. The number of data bits assigned to one tag bit is termed as *Encoding Granularity* ($Gran_{static}$). For each partition, if the number of bit-flips between the new and old data is more than half of $Gran_{static}$, then the data bits in the partition are written in inverted form. It is indicated by setting the corresponding tag bit for the partition as 1. Else, the tag bit is set as 0, and the bits are written as it is (FNW is explained in detail in Section 4.3.2.1). FNW bounds the maximum bit-flips to half of the number of bits in the partition. FNW shows more reduction in bit-flips for finer (smaller) $Gran_{static}$. The reduction in bit-flips for $Gran_{static}$= 64, 32

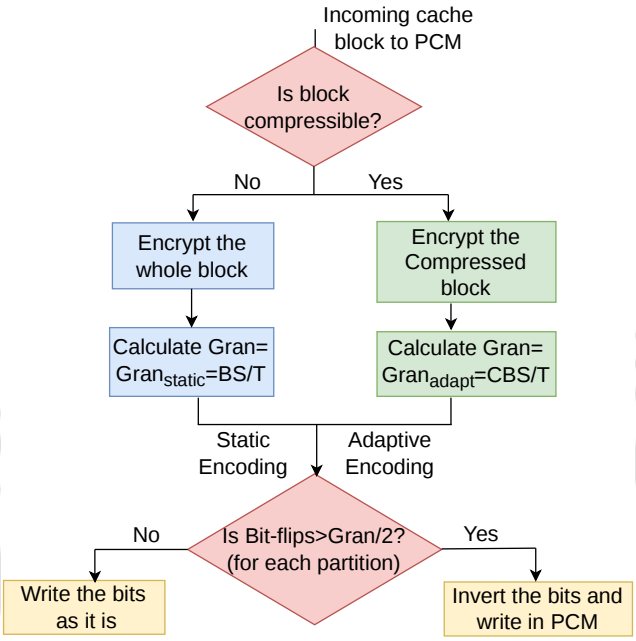


Figure 6.10: *Flowchart of CADEN (BS : Uncompressed block size, CBS: Compressed block size, T : Total tag bits)*

and 16 are 31%, 36% and 40% respectively, indicating more reduction for finer $Gran_{static}$ (as shown in Table 6.3). But, to achieve finer $Gran_{static}$, more tag bits/block are needed, which increases the storage overhead.

6.5.1 Proposed Methodology : CADEN

6.5.1.1 Motivation

COMF reduces the size of the incoming cache blocks to PCM. Compressed blocks reduce bit-flips upon writing in PCM. Bit-flips could be reduced further by encoding the compressed blocks at a finer granularity. Since compression reduces effective data to be written, the tag bits dedicated for the original uncompressed block can be judiciously used by assigning them to the compressed data portion only. In such cases, the available tag bits will govern fewer data bits during encoding which leads to finer encoding granularity. As a result, the compressed blocks can be encoded at a much finer granularity compared to $Gran_{static}$ without increasing the number of tag bits. We term this encoding granularity that depends on the size of the compressed block as the adaptive encoding granularity ($Gran_{adapt}$). Fine granularity ($Gran_{adapt}$) encoding on the compressed blocks aids further reduction in bit-flips.

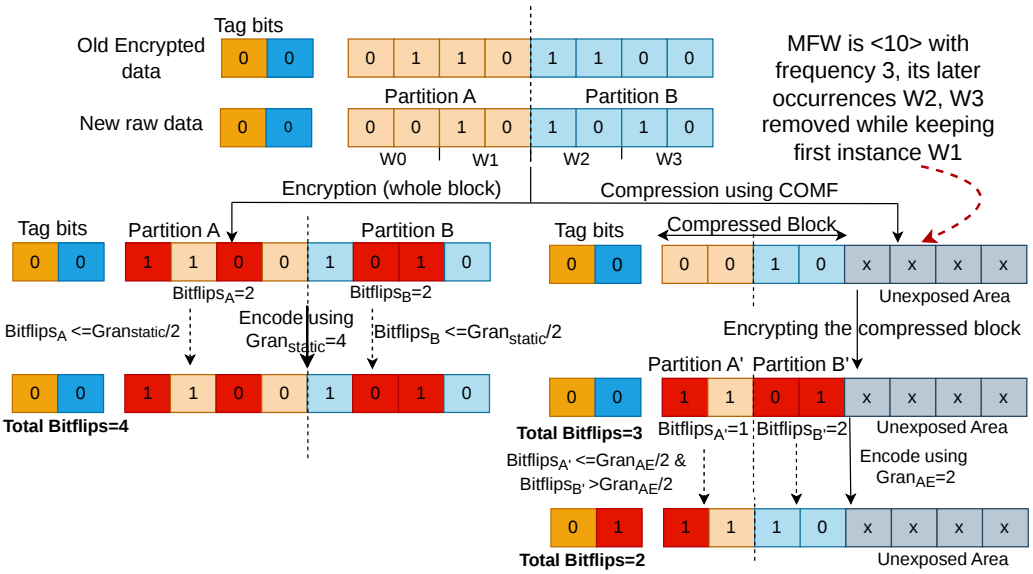


Figure 6.11: *Working Example of CADEN (bits marked red are the flipped bits compared to the old data bits)*

6.5.1.2 Working Principle of CADEN

The compressed blocks produced by COMF are encrypted using AES-based counter mode encryption [39]. CADEN applies FNW-based encoding adaptively by assigning the tag bits to the compressed data bits. Note that since the tag bits dedicated for a block is fixed, the number of partitions created by encoding is equal to the number of tag bits, where one tag bit is assigned to one partition. Since the tag bits point only to the compressed data bits, the size of the partitions reduces compared to static encoding, resulting in finer encoding granularity (we term it as adaptive encoding granularity ($Gran_{adapt}$)). Finer encoding granularity brings more reduction in bit-flips (cf. Table 6.3). $Gran_{adapt}$ maintains a direct relationship with CR where a low CR (highly compressed block) leads to more fine $Gran_{adapt}$ and vice-versa. Finally, the encoded blocks using $Gran_{adapt}$ are written in PCM. On the other hand, the fewer blocks that are left uncompressed are encoded using $Gran_{static}$. Figure 6.10 shows a flowchart of CADEN.

6.5.1.3 Working Example of CADEN

As shown in Figure 6.11, we take an incoming block of size 8 bits with 2 encoding tag bits. On the left, we show the working of FNW-based static encoding where the whole block is encrypted followed by encoding, whereas the right part shows CADEN where the block is first compressed, then encrypted, and then adaptive encoding is applied on it. If the block is encrypted directly without compression (left part of Figure 6.11), $Gran_{static}$ is 4 since

each tag bit is assigned to 4 data bits. In that case, for each of the two partitions (Partition A and B), the bit-flips is 2. Since bit-flips <= $Gran_{static}/2$, the bits are written as it is, leading to a total of 4 bit-flips.

In contrast, CADEN first compresses the block using COMF (right side of Figure 6.11) to get the content <0010>. The compressed block (<0010>) becomes <1101> upon encryption. Finally, CADEN applies adaptive encoding on this. Since only the compressed+encrypted data needs to be written, the 2 tag bits can be assigned only to the compressed+encrypted data bits. It results in a finer adaptive granularity $Gran_{adapt} = 2$ since each tag bit now governs two data bits. The bit-flips in the new partition A$'$ is 1. Since bit-flip<= $Gran_{adapt}/2$, the bits in A$'$ is written as it is with the corresponding tag bit reset to 0. On the other hand, bit-flips in partition B$'$ is 2, which is more than $Gran_{adapt}/2$. Hence, the bits in B$'$ are written in inverted form with the tag bit set as 1. It results in 2 bit-flips.

6.6 Experimental Evaluation : CADEN

6.6.1 Experimental Setup

We implement CADEN on a full system simulator GEM5 [124] integrated with NVMain [125]. The processor frequency is 2Ghz. Both L1 I/D caches are two-way associative with 32KB capacity with one cycle latency. L2 cache is 8-way associative, 8MB in size with 10 cycle latency. L2 cache is shared among the cores, while L1 caches are private to the cores. We model PCM as main memory with 4 channels, having a total capacity of 4GB. The memory controllers (dedicated per channel) use the FRFCFS scheduling policy. The latency/energy of PCM are taken as 50ns/(2.47pJ/bit) for read and 150ns/(<14.03pJ/bit (SET),19.73pJ/bit (RESET)>) ([21, 51]) for write. We take the AES encryption latency and energy as 96ns/line and 5.9nJ/128-bit block based on [145].

Workloads: We evaluate our results using multi-threaded PARSEC [116] and multi-programmed SPEC 2006 [133] benchmarks. The selected benchmarks are diverse in terms of memory intensity. We run SPEC 2006 workloads for 1B instructions after warming them up by at least 250M instructions.

6.6.2 Evaluation Metrics

We show our results in terms of a) Reduction in bit-flips b) Reduction in energy, c) Improvement in lifetime, and d) Improvement in performance (Cycles per Instruction (CPI)). We model PCM energy as the aggregate of read, write, and encryption/decryption energies. These energies mainly constitute the dynamic energy for PCM. We ignore the PCM leakage energy as it is negligible [21]. PCM shows a limited write endurance as its cells can with-

Table 6.4: *Percentage (%) Reduction in bit-flips over DCW*

Benchmark	DEUCE	BDI_{AE}	FPC_{AE}	CADEN
lbm	27	13	21	47
mcf	33	32	60	62
leslie3d	23	56	70	73
libquantum	42	41	35	45
Canneal	31	24	52	54
Dedup	17	25	30	33
Fluid	31	34	50	55
Freq	10	41	44	59
Mix1	29	46	45	48
Mix2	36	39	42	66
Mix3	38	57	61	74
Mean	**27**	**35**	**47**	**54**

stand only a limited number of writes before wearing out. We approximate lifetime as the average number of bit-flips on the cells, as mentioned in [51].

Comparison with prior work: We compare CADEN with DEUCE [39], FPC, and BDI. We support FPC and BDI with adaptive encoding (termed as FPC_{AE} and BDI_{AE}) to make an appropriate comparison with CADEN. We take a baseline where DCW is applied on the encrypted PCM.

6.6.3 Results for Bit-flips, Energy and Lifetime

Table 6.4 shows reduction in bit-flips by CADEN over DEUCE, BDI_{AE} and FPC_{AE}. CADEN outperforms DEUCE due to its high compression and the resulting finer encoding granularity, whereas DEUCE only encrypts the modified words in the blocks over epoch intervals without compressing the blocks. COMF shows low CR and high coverage compared to BDI and FPC and leads to more finer granularity when the compressed blocks are encoded (cf. Table 6.5). Many PCM cells remain unexposed when the highly compressed blocks are written to PCM. Also, compressed blocks result in finer encoding granularity which reduces bit-flips further. Table 6.6 shows comparison of bit-flips by using non-adaptive static FNW and adaptive encoding based CADEN for different tag bits. CADEN brings more reduction compared to FNW at a relatively lower storage overhead due to fine granularity encoding. For example, FNW shows 36% reduction using 16 tag bits, whereas CADEN brings 47% using only 8 tag bits. Benchmarks like mcf, leslie3d, freq have lesser CR than dedup, libquantum (CR for mcf, leslie3d, freq, dedup and libquantum are 0.45, 0.14, 0.35, 0.61, 0.56, respectively). For benchmarks with low CR, the reduction in bit-flips is more (refer to Table 6.4) compared to the other benchmarks, as they lead to highly compressed block and

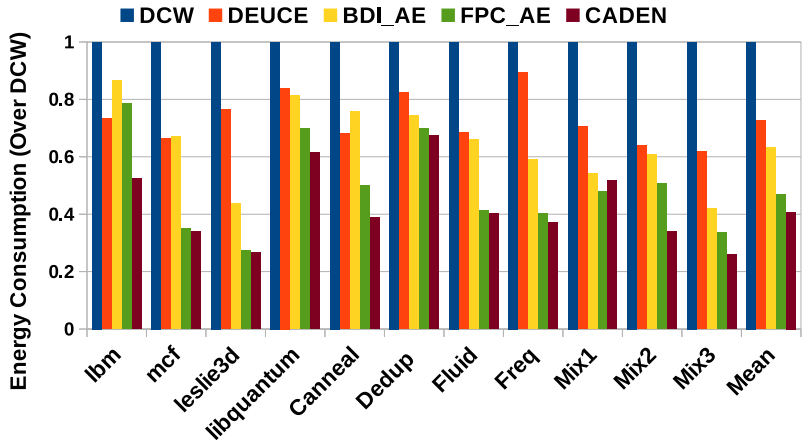


Figure 6.12: *Normalized Energy Consumption Over DCW (Less is better)*

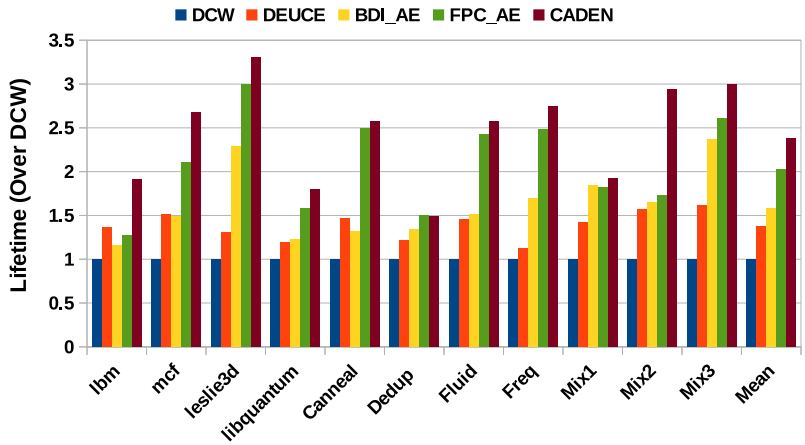


Figure 6.13: *Normalized Lifetime Over DCW (More is better)*

Table 6.5: *CR, Coverage and Adaptive Encoding Granularity for BDI, FPC and COMF(∗ : Less is better, + : More is better)*

	CR*	Cov+	$Gran^*_{static}$	$Gran^*_{adapt}$
BDI	0.41	50%	32	21
FPC	0.34	70%	32	14
COMF	0.25	79%	32	12

Table 6.6: *Comparison of Storage Overhead (SO), % Reduction in Bit-flips (RB) over DCW for FNW and CADEN for different Tag Bits (G_s : $Gran_{static}$, G_a : $Gran_{adapt}$)*

Tag Bits	SO	FNW		CADEN	
		G_s	RB	G_a	RB
8	1.56%	64	31%	25	47%
16	3.12%	32	36%	12	52%
32	6.25%	16	40%	6	55%

fine granularity encoding. Bit-flip reduction also minimizes energy consumption as writes hold a dominant part in the energy consumption in PCM. It also results in an enhanced lifetime of PCM. The improvement in <energy, lifetime> shown by DEUCE, BDI_{AE}, FPC_{AE} and CADEN are <27%, 1.37x>, <36%, 1.57x>, <53%, 2.03x> and <59%, 2.38x> over DCW, respectively (Figures 6.12 and 6.13). Reduction in worst-case bit-flips corresponding to the hottest location is 9% for CADEN compared to 8%, 7%, 3% for FPC_{AE}, BDI_{AE} and DEUCE over DCW, respectively.

6.6.4 Results for Performance

A PCM block is written over multiple slots [39] due to limited write current. Reduction in bit-flips reduces the write service time by reducing the write slots. It improves performance due to reduction in costly PCM write latency. Hence, CADEN shows better performance than DEUCE (which does not use compression), and FPC and BDI that offer lesser compression. In particular, the improvement in CPI shown by DEUCE, BDI_{AE}, FPC_{AE} and CADEN are 6%, 12%, 9% and 16%, respectively (shown in Figure 6.14).

6.6.5 Overhead Analysis

1. Storage Overhead. CADEN needs 17 bits of meta-data (1 compression bit and 16 encoding tag bits) per block. Hence, storage overhead is $(17/512) \times 100\% = 3.3\%$ only.
2. Hardware Overhead. The compressor/decompressor latency of COMF are 3.5ns and 2.65ns, respectively. The design was implemented using Verilog HDL and synthesized using Genus synthesis solution from cadence at 15nm technology node. The latency involved in

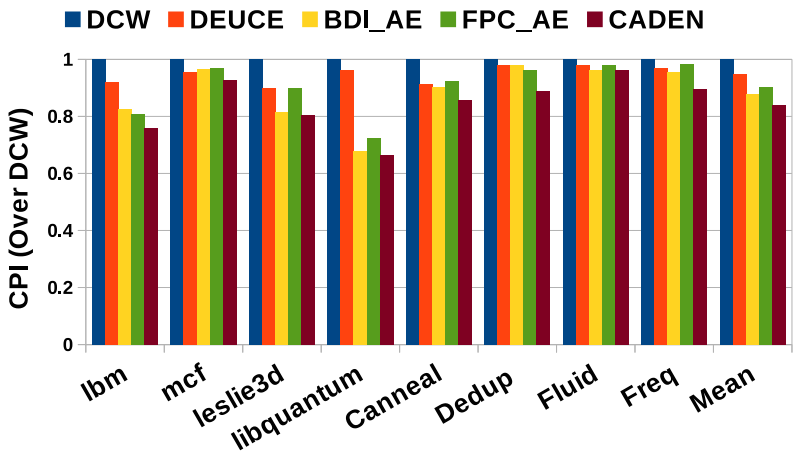


Figure 6.14: *Normalized CPI Over DCW (Less is better)*

determining $Gran_{adapt}$ is negligible as the compressed block size can be calculated using the compression meta-data. The latencies are very small compared to memory latency, incurring a negligible impact on the performance.

6.7 Summary

Non-volatile Memories are considered as suitable candidates for replacing DRAM in main memories. Approaches to secure NVM contents using encryption causes degradation of memory lifetime due to encryption-led bit-flips. In this chapter, we propose two techniques called CoSeP and CADEN that explore the benefits of compression and encoding to reduce bit-flips in encrypted PCM-based main memories. CoSeP offers an integrated approach by combining three compression techniques FPC, BDI, and COMF, to deliver optimal CR and *Coverage*. CoSeP achieves 74 % and 65% reduction in bit-flips and energy and 69% improvement in PCM lifetime over DCW. In contrast, CADEN combines the effect of our word-level compression technique COMF with an adaptive encoding approach to reduce bit-flips in encrypted PCM. High compressibility and fine granularity encoding enable CADEN to outperform existing techniques FPC_{AE}, BDI_{AE} and DEUCE by 10%, 21% and 27% in reducing bit-flips. Hence, exploring new direction using compression and encoding can assist in finding solutions to improve lifetime of NVMs in presence of encryption.

7 Conclusion

This thesis aims to improve the lifetime of the emerging non-volatile memories so that they can be utilized more effectively as a mainstream main memory standard. In order to achieve this goal, we focused our efforts in the following directions: 1) Reducing expensive writes to NVMs at the *block* and *bit* level, and distributing the writes evenly within memory blocks, 2) Providing security to the NVM data against confidentiality-based attacks using encryption and lessening the impact of encryption-driven bit-flip spikes on NVM lifetime. Towards proposing solutions for the first direction, we propose write-back reduction schemes that store the performance critical blocks evicted from the LLC in a small size victim cache. Based on *Block placement* and *Replacement policy* of the victim cache, we propose two techniques that reduce write-backs to the PCM component of a DRAM-PCM hybrid main memory. We also provide strategies to reduce bit-flipping write activities in PCM-based main memory. Towards this, we have developed a compression technique based on data similarity within the words of the memory blocks and an adaptive encoding approach that operate on the compressed data. Furthermore, our wear leveling approach evenly distributes the bit-flips within the memory lines to improve PCM lifetime further. On the other hand, for proposing solutions for the second direction, we propose a partial encryption scheme called Pop-Crypt that partially encrypts the memory blocks by eliminating encryption of the frequently appearing data words inside the blocks. This scheme reduces bit-flips initiated by the diffusion property of encryption. We also offer new avenues based on the efficient amalgamation of the existing compression algorithms with our proposed compression technique and encoding techniques to reduce bit-flips in encrypted memories, thereby improving their lifetime. This chapter sums up all the proposed contribution of this dissertation along

with the future directions for research.

7.1 Summary of Contributions

1. Reducing PCM write-backs using selective victim caching, block placement and refined replacement policy : In order to reduce the write-backs to the PCM component of DRAM-PCM based hybrid main memories, we propose two techniques called VCRP and PPVC that store the evicted blocks from the LLC in a small victim cache associated with the LLC. Keeping in view the limited capacity of the victim cache, we use the concept of reuse distance to keep only the critical blocks in the victim cache. We define the notion of criticality of the blocks based on the history of short reuse distance usage during their residency in the LLC. When a block is accessed frequently with short distance, it means that it has been used frequently enough during its LLC residency to have the potential to be used again in future memory references.

Our proposed policies keep such critical blocks inside the victim cache rather than writing them back to the main memory after eviction from LLC. Furthermore, to reduce PCM write-backs to hybrid memories further, VCRP and PPVC work on the replacement policy and block placement policy of the victim cache. Specifically, VCRP aims to retain the PCM blocks for a longer duration than the DRAM blocks in the victim cache by preferentially evicting the DRAM blocks over the PCM blocks. In contrast, PPVC logically partitions the victim cache, maintaining a smaller space for the DRAM blocks and a relatively larger space for the PCM blocks. While allocating larger space for the PCM blocks helps in retaining the critical PCM blocks in the victim cache, storing critical DRAM blocks in the victim cache gives performance benefits, as their early eviction can degrade the system performance. On average, <VCRP, PPVC> reduce PCM write-backs by <11.27%, 10.82%>, improve performance by <5.8%, 5.65%> and reduces energy consumption by <11.73%, 11.52%> over baseline.

2. Improving PCM lifetime using Compression, Adaptive Encoding and Stride-based Intra-line Wear Leveling : In this contribution, we have proposed an integrated approach using compression, data encoding, and intra-line wear leveling to reduce and uniformly distribute the bit-flips in the PCM cells. In that direction, we have developed a compression technique called COMF that exploits the abundant word-level data similarity within the data blocks. In particular, it compresses the blocks by removing the repeated instances of the most frequently occurring words inside the block. COMF provides a fine compression ratio and high coverage. On top of the compressed blocks generated by COMF, we apply FNW-based encoding adaptively by assigning the encoding tag bits to the compressed data bits only. It results in finer encoding granularity and reduces bit-flips to a

greater extent. Finally, we propose an intra-line wear leveling technique that periodically shifts the writing position (determined using a stride distance) of the compressed data to distribute the bit-flips uniformly within the cells of the memory lines. The integrated approach of compression, encoding, and wear leveling is termed SWEL-COFAE. Experimental results show that SWEL-COFAE reduces bit-flips by 59%; reduces energy consumption by 61% and improves lifetime by 101% over the baseline DCW [42] technique.

3. Improving lifetime while securing NVMs via Encryption using Partial Encryption Scheme : Non-volatility feature of the NVMs leads to potential security threats related to data stealing, as the sensitive data remains persistent even after the system is powered off. Encryption is a standard solution to protect the data in NVMs due to its high obfuscation. However, most of the standard encryption algorithms put high randomization in the generated encrypted data, leading to enormous bit-flips when the data are written in NVM in encrypted form. This is termed as the *Avalanche Effect* of the encryption algorithms.

In this contribution, we have proposed a technique called Pop-Crypt that eliminates the encryption of the frequently occurring data words that appear across several memory blocks. The frequent words are termed as *Popular words* and are stored in a table called *Popular Word Table* (PWT) inside the memory controller. Indices to the corresponding popular word in the PWT are maintained in the respective word position of the blocks to help during the de-encryption process. It reduces bit-flips due to encryption and improves lifetime of the encrypted PCMs. Pop-Crypt reduces bit-flips by 38%; reduces energy consumption by 22%; and improves lifetime by 29%, over baseline, respectively.

4. Exploring Avenues related to Compression and Encoding to reduce bit-flip in Encrypted PCM : In this contribution, we continue to find new strategies to deal with the trade-off between security provisioning via encryption and increased bit-flip as a result of encryption. We find that existing compression and encoding techniques can be utilized to build new approaches to reduce bit-flip in encrypted PCMs. Accordingly, we propose two techniques called CoSeP and CADEN based on the efficient integration of compression and encoding techniques.

The first technique, CoSeP integrates the existing compression techniques, FPC [89], BDI [90], and our proposed technique COMF. It compares the sizes of the two smallest blocks generated by these techniques. If the size differs by a wide margin, CoSeP selects the smallest block. In contrast, if the blocks differ only by a small margin, the bit-flips due to the two smallest blocks are computed, and the block that leads to minimum bit-flip is selected. The rationale behind computing bit-flip in this case is that similar-sized compressed blocks can result in highly dis-similar bit-flips upon encryption based on the content of the compressed data blocks. CoSeP optimizes compression ratio and coverage

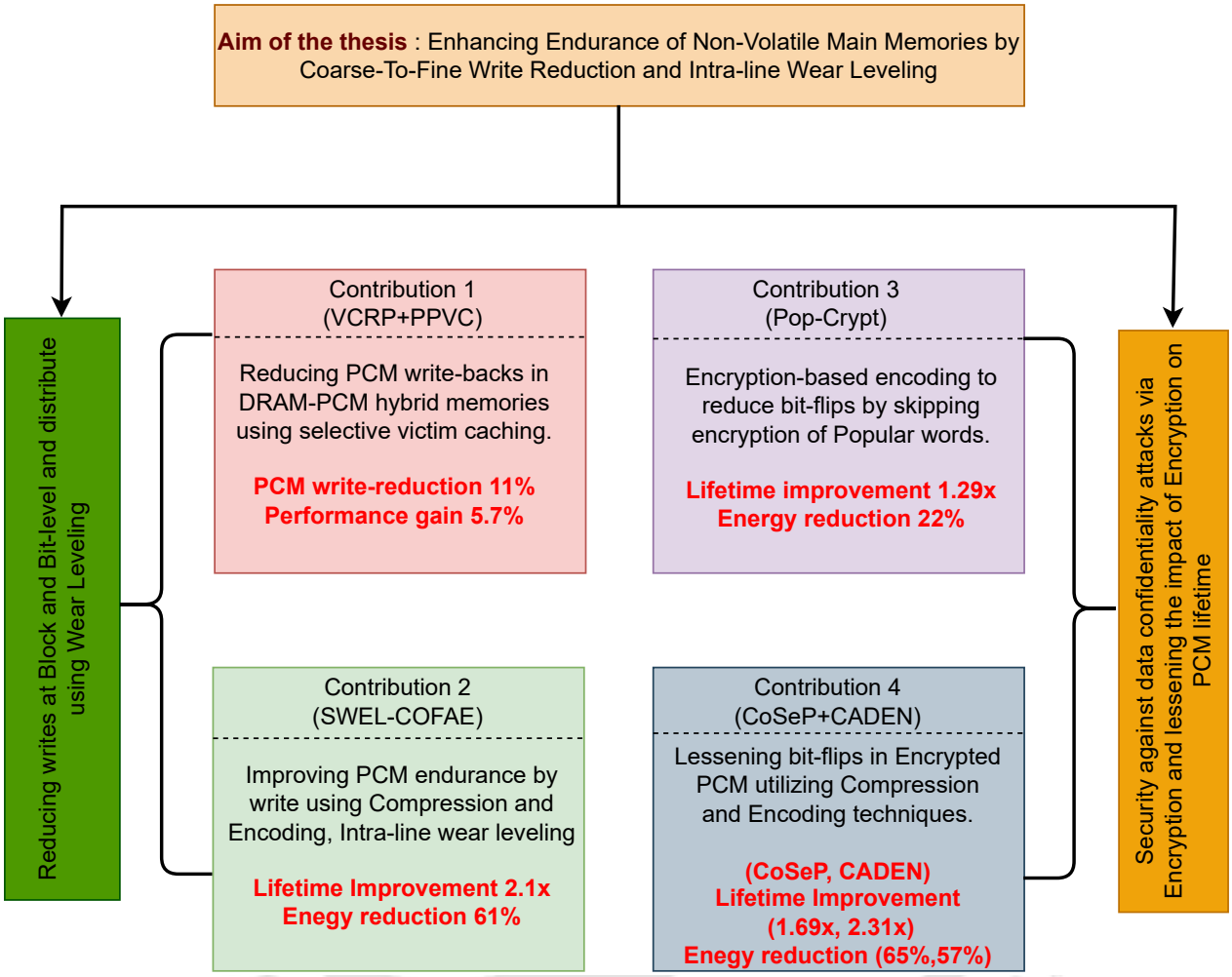


Figure 7.1: *Summary of the Thesis Contributions*

more compared to the individual techniques FPC, BDI and COMF. It achieves 74% and 65% reduction in bit-flips and energy over DCW, respectively.

The second technique, CADEN applies adaptive encoding on the compressed (generated by our proposed technique COMF) and encrypted data before writing the blocks to PCM. In this process, as the compressed block is encoded by utilizing all the encoding tag bits, it results in finer encoding granularity and achieves more reduction in bit-flips. Due to small compression ratio of COMF, the granularity of encoding is finer in the case of CADEN compared to the techniques that apply encoding on top of compressed blocks generated by FPC and BDI. CADEN achieves 52% and 51% reduction in bit-flips and energy over DCW.

Table 7.1: *Variation of DRAM, PCM miss rate and reduction in Write-back when VC is associated with LLC of sizes 4MB, 8MB and 16MB*

		%Reduction in DRAM miss rate	%Reduction in PCM miss rate	%Reduction in PCM write-back
4MB	VCRP	17.5%	17.75%	11.13%
	PPVC	15.11%	16.5%	11%
8MB	VCRP	14.9%	15.4%	9.52%
	PPVC	10.02%	14.03%	9.8%
16MB	VCRP	8.2%	11.21%	7.1%
	PPVC	7%	9%	6.8%

Table 7.2: *Reduction in Write-backs and bit-flips for SWEL-COFAE, Pop-Crypt, CADEN and CoSeP for LLC sizes 8MB and 16MB compared to 4MB LLC*

LLC Configuration	Write-back reduction (%)	Bit-flip Reduction (%) Compared to LLC = 4MB			
		SWEL	Pop-Crypt	CADEN	CoSeP
LLC = 8MB	25%	6%	20%	22%	21%
LLC = 16MB	58%	17%	58%	52%	54%

7.2 Impacts of using Bigger LLCs

Table 7.1 shows the reduction in DRAM, PCM miss rate and reduction in PCM write-back (over baseline which does not include victim cache) when victim cache is associated with bigger LLCs. It is evident from the table that our proposed techniques of selective victim caching becomes less effective for larger LLCs. It is because, for smaller LLC, the conflict misses will be high. Since conflict misses lead to eviction of critical blocks from the LLC, victim cache can provide better support by keeping such blocks for future reuse. But, for larger LLCs, eviction of the critical block will be less and selective victim caching turns out to be less effective.

Table 7.2 shows the effect of having larger LLCs for the proposed fine-grained approaches (SWEL-COFAE, Pop-Crypt, CADEN and CoSeP). As the LLC becomes larger, write-backs coming to PCM-based main memory reduce. As a result, the amount of incoming blocks to be compressed/encrypted also gets reduced which leads to reduced bit-flips compared to smaller LLC (4MB). It also lessens the latency and power consumption of compression and encryption. Table 7.2 reports the reduction in write-backs and bit-flips achieved for bigger LLCs compared to LLC having a size of 4MB for our proposed policies. It is evident from the table that along with write-backs, bit-flips reduces proportionally for 8MB and 16MB LLC compared to 4MB LLC. In other words, larger LLCs nicely supports our proposed fine-grained policies to gain better improvement in lifetime by reducing the load on the compression/encryption engines. Among the fine-grained policies, Pop-Crypt, CADEN and CoSeP show more reduction in bit-flips compared to SWEL-COFAE as they are applied on encrypted PCM where bit-flips gets enormously spiked up due to the diffusion principle of encryption.

7.3 Scope for Future Work

The contributions of this thesis can be extended in several ways. Some of the possible future research directions are listed below.

- In this thesis, we have proposed compression and encoding techniques for SLC PCM. In the future, the techniques can be extended for MLC PCM, where the endurance and energy consumption issues are more complicated.
- In chapter 4, we have proposed an intra-line wear leveling technique that periodically shifts the compressed data within the memory lines to distribute the bit-flips evenly. However, non-uniformity of writes also exists at the line level, where some lines face more writes than the other lines during system execution. This work can be extended by proposing an inter-line wear leveling technique that balances the uneven write distribution across the memory lines.
- As described in Section 2.2.2.2, overflow of the counters associated with memory blocks in counter mode encryption leads to full memory re-encryption using a new key. It stalls the system until the whole memory is re-encrypted, leading to enormous bit-flips due to full memory re-encryption. We can propose techniques to reduce the frequency of counter overflow in counter mode encryption.
- Apart from confidentiality-based attacks, we can also propose solutions to reduce the degradation of lifetime caused by write attacks where the adversaries issue frequent writes to some locations of memory with the intention of wearing them out. Early detection of such attacks and measures to be taken during an ongoing attack are the two research directions that we can work on in the future.

A Appendix

A.1 Simulation Framework

This appendix focuses on the simulation framework used in our experiments. We have used a GEM5-NVMain based co-simulation framework and used PARSEC and SPEC 2006 benchmark suite. In the following sections, we give brief overview of GEM5 and NVMain and the methodology used for integrating the two simulators. Finally, we describe the characteristics of the benchmarks used for the experiments.

A.1.1 GEM 5

GEM5 [124] is a modular event driven computing system simulation framework. Due to the highly modular structure, a user of this simulator can focus on one aspect of the code without the understanding of the entire code base. GEM5 offers diverse models of CPU, system execution mode, and memory system models.

From historical point of view, GEM5 was built by combining the best features of two existing simulators M5 and GEMS. M5 provides the diverse CPU models and multiple Instruction Set Architectures (ISA), where GEMS offers memory systems with the support of cache coherence protocols and the complete interconnection network. Below, we briefly discuss the features and the modules of M5 and GEMS.

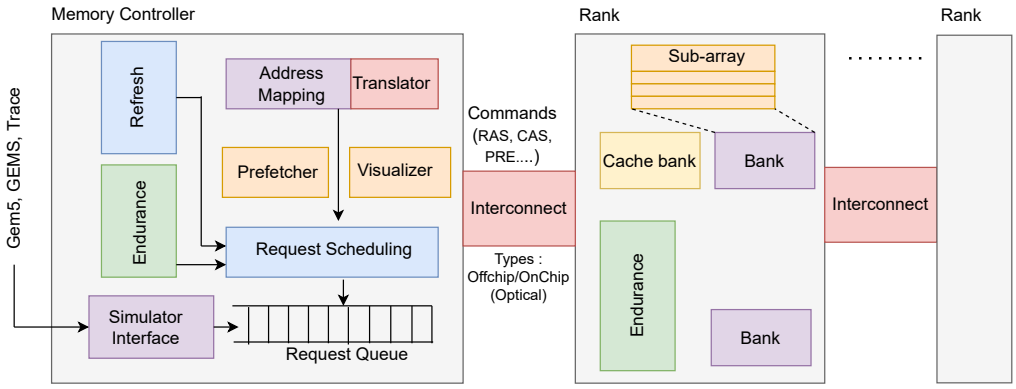


Figure A.1: *Overview of NVMain Architecture*

A.1.1.1 M5

M5 is a full system simulator that generates a complete target system or a virtual machine that runs on top of the host system. It is open source and acts as an alternative to the commercial simulators like Simics. It was developed to measure the throughput of interconnect and network protocols. M5 is flexible enough to support various CPU models, including in-order and out-of-order modes of execution. Moreover, M5 supports various ISAs like ALPHA, ARM, MIPS, Power SPARC and X86.

A.1.1.2 GEMS

GEMS comprises of two major modules: Ruby and Garnet. Ruby simulates the complete memory hierarchy of CMPs comprising L1 cache, L2 cache, memory banks and directories etc. Each component in Ruby is termed machine and is identified by its unique machine ID. The components communicate with each other using their machine IDs via the underlying Network on Chip (NoC), managed by Garnet. Garnet offers a variety of network topologies for NoC and models the real-time events for transferring packets through the NoC.

The request for a block from M5 processor is passed to the Ruby module of GEMS. If the requested block is found in the simulated first level of cache, then it is directly sent to the M5 processor and the processor continues its execution. Otherwise, the processor gets stalled until the block is delivered in the event when miss occurs. The timing dependent functional simulation is handled by Ruby.

A.1.2 NVMain

NVMain is an architectural level simulator for modeling main memory using conventional DRAM and the emerging Non-volatile memories. It integrates seamlessly with GEM5, enabling cycle-level simulation of a variety of main memory technologies in the system.

Furthermore, NVMain can also take traces as inputs to run trace-based simulation. Apart from incorporating the NVM timing parameters, NVMain also captures the unique features of NVMs by provisioning of modeling for endurance, fault recovery and MLC operation.

In NVMain, every module is created as separate objects that can be added/removed from the simulator. An overview of the architecture of NVMain is shown in Figure A.1. Every objects in the simulator captures its own timing parameters. The timing parameters related DRAM, SRAM and NVM technologies are taken from DRAM data sheets and parameters provided by CACTI and NVSIM, respectively. In order to specify the memory system hierarchy and the general configuration parameters like number of ranks, banks, rows and columns and other parameters like address mapping scheme, encoder/decoder, row buffer policies etc, configuration files are used.

NVMain provides both single bank and inter-bank timing models. For a single bank of NVM, data restoration time is taken as zero as data stored in the cells are not destroyed. Therefore, the data restoration time for NVMs can inhibit consecutive reads to different rows belonging to the same bank. For inter-bank modeling, timing parameters like tFAW (for Four Activation Window) and tRRD (Row-to-Row activation delay) are introduced. tFAW denotes the minimum interval where four activations can occur. tRRD eliminates the current hungry row activation times. These timings are applied during activation within the same as well as across different banks.

NVMain calculates energy consumed by each device, where each device consists of multiple banks. It computes total energy consumption as the summation of energy consumed by all the device. Energy consumed by the DRAM components are derived from the typical IDDx parameters for the read and write operations. In contrast, for NVM technology, NVMain takes per bit write energy from NVSIM to calculate power consumption. Endurance model in the simulator keeps tracks of the total number of bits written during simulations. Energy of the unchanged bits is subtracted from the total write energy consumption.

NVMain provides implementation of different memory controllers according to the memory technology and types. In order to model different types memory controller (DRAM, PCM), different configuration files are used where values of different parameters can be fed. For our work, we have taken memory systems with four channels. Each channel is associated with a memory controller. For creating hybrid memory (Work1), channel 0 is associated with DRAM controller while the other three channels are associated with PCM controllers. NVMain adopts PCM specific parameters from the Samsung's 2012 ISSCC [56] paper.

A.1.3 GEM5-NVMain Co-simulation Framework

In the GEM5-NVMain co-simulation framework, NVMain obtains memory requests from GEM5 at specific time instants. The requests trigger different actions in NVMain like en

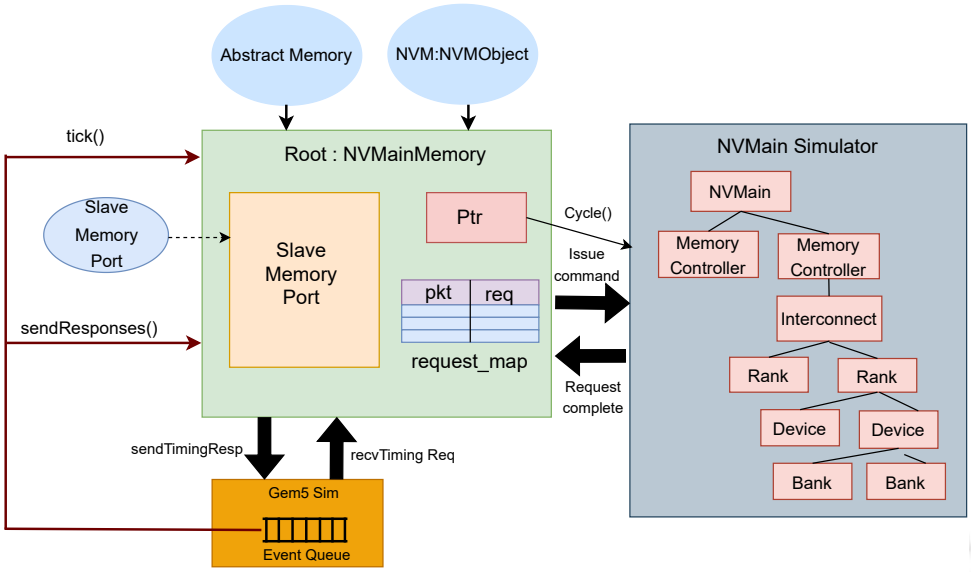


Figure A.2: *Pictorial view of Gem5-NVMain Co-Simulation Framework*

queuing and scheduling of requests, data transfer to main memory, bank latency computation and transfer data back to controller etc. These actions need to be executed before GEM5 proceeds with its subsequent tasks.

During simulation, an interfacing object called *NVMainMemory* is created from *AbstractMemory* class of GEM5 and *NVMObject* class of NVMain. The GEM5 requests appear in the form of request packet, namely *pkt*. The information of packet *pkt* is transferred onto a new NVMain specific request packet (*req*) before submitting the request to NVMain. The interfacing object stores *pkt* and the corresponding *req* in a map data structure to keep track of the issued memory requests. The information of completion of memory requests is notified to the interfacing object with the help of the function called *RequestCompleted*. Then, the corresponding *pkt* is retrieved from the map data structure and sent back to GEM5. Furthermore, *NVMainMemory*, the interfacing object and NVMain are cycled using the *tick* function, which transfers the GEM5 cycles to NVMain. Figure A.2 shows a pictorial view of the co-simulation framework of GEM5 and NVMain.

A.1.4 Timing and Power Modeling Tools : CACTI and NVSIM

GEM5-NVMain simulation framework is not able to model the latency, power and area of different memory technologies at different cache hierarchy. However, main memory related parameters are given by NVMain. In order to find the different parameters related to cache, we have used two well known simulators CACTI 6.5 [150] and NVSIM [129] to simulate the cache at device/circuit level. CACTI 6.5 is useful for modeling SRAM based caches

where as NVSIM models different emerging memory technologies like STT-RAM, ReRAM, PCM and the NAND flash etc. According to the ITRS reports [151], the caches based on SRAM and NVM memory technologies can be of three categories based on the power and performance modes : a) HP : known as High Performance cell that consumes large power with a very fast accessibility b) LSTP : Low Standby Power cells that incur low power when idle. However, they are relatively slower than the HP caches because transition from low standby power mode to active mode incur extra cycles c) LOP, known as the Low Operating Power cells consume lesser power in both the modes, standby as well the active mode. It the slowest among the three modes. Furthermore, CACTI and NVSIM supports three types of cache accesses : fast, sequential and normal. In our work, we have used the fast mode to model the caches. In this mode, the cache tag and data arrays are searched in parallel to identify for cache hit and miss. Internally, NVSIM uses the empirical modeling used by CACTI, but also includes several features which are not present in CACTI. We summarize some of the features shown by NVSIM below.

- It facilitates the user to model their memory cell configuration.
- It gives various design optimization options for buffer like latency, area etc.
- It facilitates the modeling of the memory banks in a bus like structure and H-tree like structure.
- It models various data sensing schemes rather than only the voltage based sensing.

A.2 Result Analysis

During the experiments, the co-simulated framework of GEM5 and NVMain logs different statistics of the running applications. The complete stats are composed of two types of stats, given by GEM5 and NVMain respectively. Some of the important information needed in this thesis are given below.

1. GEM5 Related Stats

- **Total Cycles Executed :** This metric records the summation of all the cycles executed in all the cores.
- **Total simulated instruction :** It collects the instructions executed in each core and gives the summation of all the instructions executed.
- **L1 Demand Access :** GEM5 records the demand hits/misses during access to each L1 cache bank private to a core.

- **L2 Demand Access :** GEM5 records the demand hits/misses during accesses to the shared L2 bank.

2. NVMain Related Stats

- **Memory Reads:** It gives the total number of read requests issued to the main memory. NVMain also collects the read requests to each memory sub-modules like channel, rank, bank, subarray etc.
- **Memory Writes:** It gives the total number of write requests issued to the main memory. NVMain also collects the write requests to each memory sub-modules like channel, rank, bank, subarray etc.
- **Endurance:** For endurance, we have used the byte model of the endurance module of NVMain. It gives the worst-case and average-case write counts to an individual byte of the NVM main memory.

Apart from this, we have also added some extra statistics needed for our analysis. We highlight such stats below. While the first three stats are added to the GEM5 stats, the latter stats are added to the NVMain stats.

- **DRAM Reads/Writes (GEM5 Stats):** It gives the total read and write counts to the DRAM portion of the DRAM-PCM hybrid main memory.
- **NVM Reads/writes (GEM5 Stats):** It gives the total read and write counts to the PCM portion of the DRAM-PCM hybrid main memory.
- **DRAM/NVM Miss Rate (GEM5 Stats):** It gives the miss rates of the DRAM/NVM blocks in the LLC, where the underlying main memory consists of DRAM and PCM.
- **Bit-flips (NVMain Stats):** Aggregate bit-flips in all the memory cells when a new cache line is written in NVM-based main memory.
- **Compression Ratio (NVMain Stats):** Average size of a compressed block relative to the original block size, as determined by the formula 2.5.
- **Coverage (NVMain Stats):** Percentage of compressible blocks, as determined by the formula 2.6.

A.3 Benchmarks

Benchmarks are real world applications run on the simulated architecture created by simulators. Based on the simulation results, the power and performance related parameters are evaluated for the new architecture. PARSEC, SPEC CPU 2006 and SPLASH-2 etc are popular benchmark suites used in architectural research. In this thesis, we have used multi-threaded PARSEC and multi-programmed SPEC CPU 2006 benchmark suite to determine the effectiveness of our proposed architectural solutions. The detailed description of these benchmarks is given below.

A.3.1 PARSEC

The Princeton Application Repository for Shared-Memory Computers(PARSEC) [116] suite is a collection of multi-threaded benchmarks designed specifically for the evaluation and validation of the next generation CMPs. It was developed collaboratively by Princeton university and Intel to help the research community for efficient designing of the future computing systems. It is open source and widely accepted in academic as well industrial research. PARSEC version 2.1 suite contains 12 applications, each application is parallelized and multi-threaded. The applications are selected from diverse areas of the real world like computer vision, animation physics, finance, media processing etc. Table A.1 shows the detailed description of the benchmarks. Due to multi-threading, these applications share data between the spawned threads. Table A.2 shows the data sharing and exchange behavior of the applications. Each benchmark in the PARSEC benchmark suite has their own working set and input sizes : small, medium and large. Depending on the requirement and architecture design, users can run benchmarks with the appropriate input sizes.

A.3.2 SPEC 2006

Standard Performance Evaluation Corporation (SPEC) CPU 2006 [133] is an industry standard benchmark suite developed to determine the performance of compiler, processor and the memory hierarchy. Below we describe the two types of suites of SPEC 2006 that focuses on two types of compute-intensive performance.

- **CINT 2006 benchmark suite :** The benchmarks are used to evaluate the performance of the compute-intensive integer operations. It contains 12 benchmarks and the description of the benchmarks is provided in Table A.3.
- **CFP 2006 benchmark suite:** These benchmarks are used to evaluate the performance of the compute-intensive floating point operations. It contains 17 benchmarks and the description of these benchmarks is provided in Table A.4.

Table A.1: *The Inherent Key Characteristics of PARSEC Benchmarks*

Benchmarks	Application Domain	Parallelization		Working set
		Model	Granularity	
blackscholes	Financial Analysis	data-parallel	coarse	small
bodytrack	Computer Vision	data-parallel	medium	medium
canneal	Engineering	unstructured	fine	unbounded
dedup	Enterprise Storage	pipeline	medium	unbounded
facesim	Animation	data-parallel	coarse	large
ferret	Similarity Search	pipeline	medium	unbounded
fluidanimate	Animation	data-parallel	fine	large
freqmine	Data Mining	data-parallel	medium	unbounded
streamcluster	Data Mining	data-parallel	medium	medium
swaptions	Financial Analysis	data-parallel	coarse	medium
vips	Media Processing	data-parallel	coarse	medium
x264	Media Processing	pipeline	coarse	medium

Table A.2: *The Data Usage Behavior of PARSEC Benchmarks*

Benchmarks	Data Exchange	
	Sharing	Exchange
blackscholes, swaptions	low	low
bodytrack, freqmine	high	medium
canneal, dedup, ferret, x264	high	high
facesim, fluidanimate, streamcluster, vips	low	medium

Table A.3: *Application Domains of Various CINT 2006 Benchmark Suite*

Workload	Programming Language	Application Domain
400.perlbench	C	Programming Language
401.bzip2	C	Compression
403.gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	Artificial Intelligence: Go
456.hmmer	C	Search Gene Sequence
458.sjeng	C	Artificial Intelligence: chess
462.libquantum	C	Physics / Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.xalancbmk	C++	XML Processing

SPEC 2006 suite adopts different ways to quantify the performance of computer. For example, SPECrate metric measures the number of distinct tasks a computer can complete in a definite time. In contrast, SPECspeed is a metric to determine the speed with which a computer completes a given task.

A.4 Benchmark Running Procedure

In order to evaluate the proposed architectural solutions, we have used several multi-threaded and multi-programmed benchmarks in the simulation. PARSEC benchmarks are used for multi-threading. The number of threads in each PARSEC benchmark depends on input size and the load of the program. In the PARSEC benchmark, multi-threading execution occurs within a period of time called Region Of Interest (ROI). The scanning and initialization of the variables are performed before entering ROI. Finally, the workload completes its execution as the ROI finishes after generating the output. In this section, we illustrate the execution procedure of different multi-threaded and multi-programmed benchmarks. In contrast, multi-programmed benchmarks are built by combining various SPEC2006 benchmarks. For example, we can create a mix benchmark by combining four benchmarks bzip2, mcf, milc and leslie3d for a four core CMP. The individual application run on their specific cores until completion of the specified number of instructions.

1. Running of Multi-threaded benchmarks : PARSEC benchmarks suite contains many multi-threaded applications. We run PARSEC benchmarks individually in the simulated system up to the completion of the benchmark. During the entire simulation, four

Table A.4: *Application Domains of Various CFP 2006 Benchmark Suite*

Workload	Programming Language	Application Domain
410.bwaves	Fortran	Fluid Dynamics
416.gamess	Fortran	Quantum Chemistry
433.milc	C	Physics/Quantum Chromodynamics
434.zeusmp	Fortran	Physics / CFD
435.gromacs	C, Fortran	Biochemistry / Molecular Dynamics
436.cactusADM	C, Fortran	Physics / General Relativity
437.leslie3d	Fortran	Fluid Dynamics
444.namd	C++	Biology / Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.soplex	C++	Linear Programming, Optimization
453.povray	C++	Image Ray-tracing
454.calculix	C, Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.tonto	Fortran	Quantum Chemistry
470.lbm	C	Fluid Dynamics
481.wrf	C, Fortran	Weather
482.sphinx3	C	Speech recognition

stats are dumped in the generated stats file that comprises of 1) statistics for M5 full booting process 2) statistics before entering ROI that include initialization and the spawning of the threads 3) The statistics pertaining to the ROI 4) statistics from the end of the ROI till the end of the simulation exit. The third statistics corresponding to the ROI represents the actual execution of the PARSEC applications.

2. Running of Multi-programmed benchmarks : SPEC multi-programmed benchmarks are loaded one by one and each benchmark is run for a warm up phase of 250 million instruction. The warm up phase is essential to move beyond the compulsory misses in the cache, and prepares the proposed architecture to settle properly for simulation. After warm up phase, each benchmark is run for 1 billion instruction to collect the stats necessary for analyzing the performance of the proposed architectural design.

Publications Related To Thesis

Conferences

1. **Arijit Nath** and Hemangee K. Kapoor. "CoSeP: Compression and Content-based Selection Procedure to improve lifetime of encrypted Non-Volatile Main Memories" The ACM Great Lakes Symposium on VLSI (GLSVLSI) 2022, ACM.

2. **Arijit Nath** and Hemangee K. Kapoor. "WELCOMF: wear leveling assisted compression using frequent words in non-volatile main memories", ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2020.

Journals

1. **Arijit Nath**, and Hemangee K. Kapoor. "Pop-Crypt: Identification and Management of Popular Words forEnhancing Lifetime of EnCrypted Nonvolatile Main Memories.", IEEE Transactions on Very Large Scale Integration (VLSI) Systems 30.9 (2022): 1219-1229.

2. **Arijit Nath**, and Hemangee K. Kapoor. Arijit Nath, and Hemangee K. Kapoor. "CADEN: Compression Assisted ADaptive Encoding to improve lifetime of Encrypted Non-Volatile Main Memories.", IEEE Embedded Systems Letters (2022).

3. **Arijit Nath**, and Hemangee K. Kapoor. Arijit Nath, and Hemangee K. Kapoor. "SWELCOFAE: Wear Leveling and Adaptive Encoding Assisted Compression of Frequent Words in Non-Volatile Main Memories.", IEEE Transactions on Computers, vol. 71, no. 9, pp. 2263-2276, 1 Sept. 2022, doi: 10.1109/TC.2021.3126156.

4. **Arijit Nath**, Sukarn Agarwal, and Hemangee K. Kapoor. "Reuse distance-based victim cache for effective utilisation of hybrid main memory system.", ACM Transactions on Design Automation of Electronic Systems (TODAES) 25.3 (2020): 1-32.

Other Publications of The Author

Conferences

1. Nishant Bharti, **Arijit Nath**, Swati Upadhyay and Hemangee K. Kapoor. "ZOCHEN: Compression using Zero chain elimination and encoding to improve endurance of Non-volatile Memories", IEEE International Symposium on Quality Electronic Design (ISQED), 2023 (**To appear**).

2. Swati Upadhyay, **Arijit Nath** and Hemangee K. Kapoor. "Exploiting successive identical words and differences with dynamic bases for effective compression in Non-Volatile Memories" ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2022.

3. **Arijit Nath**, Manik B. Bhosle and Hemangee K. Kapoor. "SeNonDiv: Securing Non-Volatile Memory using Hybrid Memory and Critical Data Diversion.", IEEE International Symposium on Quality Electronic Design (ISQED), 2021.

4. **Arijit Nath** and Hemangee K. Kapoor. "Write Variation Aware Cache Partitioning for Improved Lifetime in Non-volatile Caches", IEEE International Conference on Embedded Systems (VLSID), 2019.

5. Alankar V. Umdekar, **Arijit Nath**, Shirshendu Das and Hemangee K. Kapoor. "Dynamic Thermal Management by Using Task Migration in Conjunction with Frequency Scaling for Chip Multiprocessors", IEEE International Conference on Embedded Systems (VLSID), 2018.

References

[1] Gordon E Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998. [Pg.1]

[2] Robert H Dennard, Fritz H Gaensslen, Hwa-Nien Yu, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of solid-state circuits*, 9(5):256–268, 1974. [Pg.1]

[3] David W Wall. Limits of instruction-level parallelism. In *Proceedings of the fourth international conference on Architectural support for programming languages and operating systems*, pages 176–188, 1991. [Pg.1]

[4] Jeff Parkhurst, John Darringer, and Bill Grundmann. From single core to multi-core: preparing for a new exponential. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 67–72, 2006. [Pg.1], [Pg.2]

[5] Intel xeon phi processors product specifications, . URL `https://ark.intel.com/content/www/us/en/ark/products/series/75557/intel-xeon-phi-processors.html`. [Pg.1]

[6] URL `https://www.amd.com/en/processors/epyc-server-cpu-family`. [Pg.1]

[7] High performance scalable sustainable. URL `https://amperecomputing.com/processors/ampere-altra/`. [Pg.1]

[8] Lavanya Subramanian, Vivek Seshadri, Yoongu Kim, Ben Jaiyen, and Onur Mutlu. MISE: Providing performance predictability and improving fairness in shared main memory systems. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 639–650. IEEE, 2013. [Pg.1]

[9] Reetuparna Das, Rachata Ausavarungnirun, Onur Mutlu, Akhilesh Kumar, and Mani Azimi. Application-to-core mapping policies to reduce memory system interference in multi-core systems. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 107–118. IEEE, 2013. [Pg.1]

REFERENCES

[10] Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu. The application slowdown model: Quantifying and controlling the impact of inter-application interference at shared caches and main memory. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 62–75. IEEE, 2015. [Pg.1]

[11] JEDEC Solid State Technology Assn. JESD79-3F: DDR3 SDRAM Standard, 2012. . [Pg.1]

[12] JEDEC Solid State Technology Assn. JESD79-4B: DDR4 SDRAM Standard, 2017. . [Pg.1]

[13] N.P. Jouppi and S.J.E. Wilton. Tradeoffs in two-level on-chip caching. In *Proceedings of 21 International Symposium on Computer Architecture*, pages 34–45, 1994. doi: 10.1109/ISCA.1994.288163. [Pg.2]

[14] Rajeev Balasubramonian, Norman P Jouppi, and Naveen Muralimanohar. Multi-core cache hierarchies. *Synthesis Lectures on Computer Architecture*, 6(3):1–153, 2011. [Pg.2]

[15] Randal E Bryant. Data-intensive supercomputing: The case for DISC. 2007. [Pg.2]

[16] Donghyuk Lee, Farhad Hormozdiari, Hongyi Xin, Faraz Hach, Onur Mutlu, and Can Alkan. Fast and accurate mapping of complete genomics reads. *Methods*, 79:3–10, 2015. [Pg.2]

[17] Divyakant Agrawal, Philip Bernstein, Elisa Bertino, Susan Davidson, Umeshwas Dayal, Michael Franklin, Johannes Gehrke, Laura Haas, Alon Halevy, Jiawei Han, et al. Challenges and opportunities with big data 2011-1. 2011. [Pg.2]

[18] Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Michael Kistler, and Tom W. Keller. Energy management for commercial servers. *Computer*, 36 (12):3948, dec 2003. ISSN 0018-9162. doi: 10.1109/MC.2003.1250880. URL `https://doi.org/10.1109/MC.2003.1250880`. [Pg.2]

[19] Uksong Kang, Hak-Soo Yu, Churoo Park, Hongzhong Zheng, John Halbert, Kuljit Bains, S Jang, and Joo Sun Choi. Co-architecting controllers and DRAM to enhance DRAM process scaling. In *The memory forum*, volume 14, 2014. [Pg.2]

[20] Jack A Mandelman, Robert H Dennard, Gary B Bronner, John K DeBrosse, Rama Divakaruni, Yujun Li, and Carl J Radens. Challenges and future directions for the scaling of dynamic random-access memory (DRAM). *IBM Journal of Research and Development*, 46(2.3):187–212, 2002. [Pg.2]

[21] Benjamin C Lee, Engin Ipek, Onur Mutlu, and Doug Burger. Architecting phase change memory as a scalable DRAM alternative. In *Proceedings of the 36th annual international symposium on Computer architecture*, pages 2–13, 2009. [Pg.2], [Pg.5], [Pg.23], [Pg.25], [Pg.92], [Pg.140], [Pg.148]

[22] H-S Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010. [Pg.2]

[23] Benjamin C Lee, Engin Ipek, Onur Mutlu, and Doug Burger. Phase change memory architecture and the quest for scalability. *Communications of the ACM*, 53(7):99–106, 2010. [Pg.2]

[24] Moinuddin K Qureshi, Vijayalakshmi Srinivasan, and Jude A Rivers. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the 36th annual international symposium on Computer architecture*, pages 24–33, 2009. [Pg.2], [Pg.5]

[25] Emre Kültürsay, Mahmut Kandemir, Anand Sivasubramaniam, and Onur Mutlu. Evaluating STT-RAM as an energy-efficient main memory alternative. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 256–267. IEEE, 2013. [Pg.2]

[26] Moinuddin K Qureshi, Sudhanva Gurumurthi, and Bipin Rajendran. Phase change memory: From devices to systems. *Synthesis Lectures on Computer Architecture*, 6 (4):1–134, 2011. [Pg.2], [Pg.5]

[27] Rachata Ausavarungnirun, Kevin Kai-Wei Chang, Lavanya Subramanian, Gabriel H Loh, and Onur Mutlu. Staged memory scheduling: Achieving high performance and scalability in heterogeneous systems. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 416–427. IEEE, 2012. [Pg.2]

[28] Eric S Chung, Peter A Milder, James C Hoe, and Ken Mai. Single-chip heterogeneous computing: Does the future include custom logic, FPGAs, and GPGPUs? In *2010 43rd annual IEEE/ACM international symposium on microarchitecture*, pages 225–236. IEEE, 2010. [Pg.2]

[29] Stijn Eyerman and Lieven Eeckhout. Modeling critical sections in amdahl's law and its implications for multicore design. In *Proceedings of the 37th annual international symposium on Computer architecture*, pages 362–370, 2010. [Pg.2]

[30] José A Joao, M Aater Suleman, Onur Mutlu, and Yale N Patt. Utility-based acceleration of multithreaded applications on asymmetric cmps. *ACM SIGARCH Computer Architecture News*, 41(3):154–165, 2013. [Pg.2]

[31] Anteneh Gebregiorgis, Hoang Anh Du Nguyen, Jintao Yu, Rajendra Bishnoi, Mottaqiallah Taouil, Francky Catthoor, and Said Hamdioui. A survey on memory-centric computer architectures. *J. Emerg. Technol. Comput. Syst.*, jun 2022. ISSN 1550-4832. doi: 10.1145/3544974. URL `https://doi.org/10.1145/3544974`. Just Accepted. [Pg.2]

[32] Saugata Ghose, Amirali Boroumand, Jeremie S Kim, Juan Gómez-Luna, and Onur Mutlu. Processing-in-memory: A workload-driven perspective. *IBM Journal of Research and Development*, 63(6):3–1, 2019. [Pg.2]

[33] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, Stephen Heil, Prerak Patel, Adam Sapek, Gabriel Weisz, Lisa Woods, Sitaram Lanka, Steven K. Reinhardt, Adrian M. Caulfield, Eric S. Chung, and Doug Burger. A configurable cloud-scale DNN processor for real-time AI, year=2018, volume=, number=, pages=1-14, doi=10.1109/ISCA.2018.00012. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. [Pg.2]

[34] Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. Profiling a warehouse-scale computer. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 158–169, 2015. [Pg.2]

[35] Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. A scalable processing-in-memory accelerator for parallel graph processing. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 105–117, 2015. [Pg.3]

[36] Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. Pim-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pages 336–348. IEEE, 2015. [Pg.3]

[37] Mingyu Gao, Grant Ayers, and Christos Kozyrakis. Practical near-data processing for in-memory analytics frameworks. In *2015 International Conference on Parallel Architecture and Compilation (PACT)*, pages 113–124. IEEE, 2015. [Pg.3]

[38] Kevin Lim, Jichuan Chang, Trevor Mudge, Parthasarathy Ranganathan, Steven K Reinhardt, and Thomas F Wenisch. Disaggregated memory for expansion and sharing

in blade servers. *ACM SIGARCH computer architecture news*, 37(3):267–278, 2009. [Pg.xiv], [Pg.3]

[39] Vinson Young, Prashant J. Nair, and Moinuddin K. Qureshi. DEUCE: Write-efficient encryption for non-volatile memories. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, page 3344, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450328357. doi: 10.1145/2694344.2694387. URL `https://doi.org/10.1145/2694344.2694387`. [Pg.5], [Pg.13], [Pg.14], [Pg.27], [Pg.28], [Pg.37], [Pg.39], [Pg.41], [Pg.42], [Pg.94], [Pg.108], [Pg.111], [Pg.123], [Pg.124], [Pg.127], [Pg.131], [Pg.132], [Pg.134], [Pg.140], [Pg.147], [Pg.149], [Pg.151]

[40] Shivam Swami, Joydeep Rakshit, and Kartik Mohanram. SECRET: Smartly encrypted energy efficient non-volatile memories. In *Proceedings of the 53rd Annual Design Automation Conference*, pages 1–6, 2016. [Pg.5], [Pg.13], [Pg.14], [Pg.27], [Pg.37], [Pg.38], [Pg.39], [Pg.111], [Pg.123], [Pg.124], [Pg.132]

[41] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. A durable and energy efficient main memory using phase change memory technology. *ACM SIGARCH computer architecture news*, 37(3):14–23, 2009. [Pg.5], [Pg.29], [Pg.40], [Pg.41], [Pg.42]

[42] Byung-Do Yang, Jae-Eun Lee, Jang-Su Kim, Junghyun Cho, Seung-Yun Lee, and Byoung-Gon Yu. A low power phase-change random access memory using a data-comparison write scheme. In *2007 IEEE International Symposium on Circuits and Systems*, pages 3014–3017. IEEE, 2007. [Pg.6], [Pg.13], [Pg.14], [Pg.27], [Pg.29], [Pg.34], [Pg.80], [Pg.94], [Pg.123], [Pg.124], [Pg.135], [Pg.140], [Pg.155]

[43] Sangyeun Cho and Hyunjin Lee. Flip-N-Write: A simple deterministic technique to improve pram write performance, energy and endurance. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 347–357, 2009. [Pg.12], [Pg.15], [Pg.27], [Pg.34], [Pg.36], [Pg.37], [Pg.80], [Pg.85], [Pg.86], [Pg.94], [Pg.96], [Pg.108], [Pg.123], [Pg.124], [Pg.135]

[44] Jie Xu, Dan Feng, Yu Hua, Wei Tong, Jingning Liu, Chunyan Li, Gaoxiang Xu, and Yiran Chen. Adaptive granularity encoding for energy-efficient non-volatile main memory. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019. [Pg.6], [Pg.12], [Pg.13], [Pg.14], [Pg.35], [Pg.37], [Pg.86], [Pg.94], [Pg.106], [Pg.108], [Pg.109]

[45] Norman P Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. *ACM SIGARCH Computer Architecture News*, 18(2SI):364–373, 1990. [Pg.10]

[46] Fang Zhou, Song Wu, Youchuang Jia, Xiang Gao, Hai Jin, Xiaofei Liao, and Pingpeng Yuan. VAIL: A victim-aware cache policy to improve nvm lifetime for hybrid memory system. *Parallel Computing*, 87:70–76, 2019. [Pg.11], [Pg.31], [Pg.32], [Pg.47], [Pg.60], [Pg.71]

[47] Deshan Zhang, Lei Ju, Mengying Zhao, Xiang Gao, and Zhiping Jia. Write-back aware shared last-level cache management for hybrid main memory. In *Proceedings of the 53rd Annual Design Automation Conference*, pages 1–6, 2016. [Pg.11], [Pg.32], [Pg.33], [Pg.34], [Pg.45], [Pg.47], [Pg.60]

[48] Adam N Jacobvitz, Robert Calderbank, and Daniel J Sorin. Coset coding to extend the lifetime of memory. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 222–233. IEEE, 2013. [Pg.12], [Pg.15], [Pg.35], [Pg.36], [Pg.37], [Pg.39]

[49] David B Dgien, Poovaiah M Palangappa, Nathan A Hunter, Jiayin Li, and Kartik Mohanram. Compression architecture for bit-write reduction in non-volatile memory technologies. In *2014 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 51–56. IEEE, 2014. [Pg.13], [Pg.94], [Pg.108], [Pg.109]

[50] Jie Xu, Dan Feng, Yu Hua, Wei Tong, Jingning Liu, and Chunyan Li. Extending the lifetime of NVMs with compression. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1604–1609. IEEE, 2018. [Pg.13], [Pg.92], [Pg.94], [Pg.108], [Pg.109]

[51] Dan Feng, Jie Xu, Yu Hua, Wei Tong, Jingning Liu, Chunyan Li, and Yiran Chen. A low-overhead encoding scheme to extend the lifetime of nonvolatile memories. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10): 2516–2529, 2019. [Pg.13], [Pg.94], [Pg.95], [Pg.135], [Pg.140], [Pg.144], [Pg.148], [Pg.149]

[52] S-W Chung, Tatsuya Kishi, Jung Woo Park, Masatoshi Yoshikawa, Kyung-Seok Park, Toshihiko Nagase, K Sunouchi, H Kanaya, GC Kim, K Noma, et al. 4Gbit density STTMRAM using perpendicular MTJ realized with compact cell structure. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 27–1. IEEE, 2016. [Pg.21]

[53] Oleg Golonzka, J-G Alzate, U Arslan, M Bohr, P Bai, J Brockman, B Buford, C Connor, N Das, B Doyle, et al. MRAM as embedded non-volatile memory solution for 22FFL FinFET technology. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 18–1. IEEE, 2018. [Pg.21]

[54] YJ Song, JH Lee, SH Han, HC Shin, KH Lee, K Suh, DE Jeong, GH Koh, SC Oh, JH Park, et al. Demonstration of highly manufacturable STT-MRAM embedded in 28nm logic. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 18–2. IEEE, 2018. [Pg.21]

[55] Kwang-Jin Lee, Beak-Hyung Cho, Woo-Yeong Cho, Sangbeom Kang, Byung-Gil Choi, Hyung-Rok Oh, Chang-Soo Lee, Hye-Jin Kim, Joon-Min Park, Qi Wang, et al. A 90 nm 1.8 v 512 Mb diode-switch PRAM with 266 mb/s read throughput. *IEEE Journal of Solid-State Circuits*, 43(1):150–162, 2008. [Pg.22]

[56] Youngdon Choi, Ickhyun Song, Mu-Hui Park, Hoeju Chung, Sanghoan Chang, Beakhyoung Cho, Jinyoung Kim, Younghoon Oh, Duckmin Kwon, Jung Sunwoo, et al. A 20nm 1.8 v 8Gb PRAM with 40mb/s program bandwidth. In *2012 IEEE International Solid-State Circuits Conference*, pages 46–48. IEEE, 2012. [Pg.22], [Pg.161]

[57] Intel,STMicroelectronics deliver Industry's First Phase Change Memory Prototypes [online]. Available : https://web.archive.org/web/20080609215913/http://www.numonyx.com/en-us/about/pressroom/releases/pages/intelstdeliverfirstpcmprototypes.aspx, . [Pg.22]

[58] intel launches optane memory m.2 cache ssds for consumer market [online]. Available : https://www.anandtech.com/show/11227/intel-launches-optane-memory-m2-cache-ssds-for-client-market , . [Pg.22]

[59] Saeed Rashidi, Majid Jalili, and Hamid Sarbazi-Azad. A survey on PCM lifetime enhancement schemes. *ACM Computing Surveys (CSUR)*, 52(4):1–38, 2019. [Pg.22], [Pg.25]

[60] What Happened To ReRAM? [online]. Available : https://semiengineering.com/what-happened-to-reram/". [Pg.23]

[61] Cheng Chen, Qingsong Wei, Weng-Fai Wong, and Chundong Wang. NV-Journaling: Locality-aware journaling using byte-addressable non-volatile memory. *IEEE Transactions on Computers*, 69(2):288–299, 2019. [Pg.23]

[62] Mark H Kryder and Chang Soo Kim. After hard driveswhat comes next? *IEEE Transactions on Magnetics*, 45(10):3406–3413, 2009.

[63] Xiangyu Dong, Xiaoxia Wu, Guangyu Sun, Yuan Xie, Helen Li, and Yiran Chen. Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. In *2008 45th ACM/IEEE Design Automation Conference*, pages 554–559. IEEE, 2008.

[64] Sparsh Mittal, Jeffrey S Vetter, and Dong Li. A survey of architectural approaches for managing embedded DRAM and non-volatile on-chip caches. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1524–1537, 2014.

[65] Geoffrey W Burr, Bülent N Kurdi, J Campbell Scott, Chung Hon Lam, Kailash Gopalakrishnan, and Rohit S Shenoy. Overview of candidate device technologies for storage-class memory. *IBM Journal of Research and Development*, 52(4.5):449–464, 2008.

[66] Dean L Lewis and Hsien-Hsin S Lee. Architectural evaluation of 3D stacked RRAM caches. In *2009 IEEE International Conference on 3D System Integration*, pages 1–4. IEEE, 2009. [Pg.23]

[67] Luiz E Ramos, Eugene Gorbatov, and Ricardo Bianchini. Page placement in hybrid memory systems. In *Proceedings of the international conference on Supercomputing*, pages 85–95, 2011. [Pg.25]

[68] Wei Wei, Dejun Jiang, Jin Xiong, and Mingyu Chen. HAP: Hybrid-memory-aware partition in shared last-level cache. *ACM Transactions on Architecture and Code Optimization (TACO)*, 14(3):1–25, 2017. [Pg.25], [Pg.32], [Pg.33], [Pg.34], [Pg.45], [Pg.59]

[69] Moinuddin K Qureshi, Vijayalakshmi Srinivasan, and Jude A Rivers. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the 36th annual international symposium on Computer architecture*, pages 24–33, 2009. [Pg.25], [Pg.31]

[70] Hyunsun Park, Chanha Kim, Sungjoo Yoo, and Chanik Park. Filtering dirty data in DRAM to reduce PRAM writes. In *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 319–324. IEEE, 2015. [Pg.25], [Pg.31]

[71] Muhammad Imran, Taehyun Kwon, and Joon-Sung Yang. Effective write disturbance mitigation encoding scheme for high-density PCM. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1490–1495. IEEE, 2020. [Pg.26]

[72] Bob Gleixner, Fabio Pellizzer, and Roberto Bez. Reliability characterization of phase change memory. In *2009 10th Annual Non-Volatile Memory Technology Symposium (NVMTS)*, pages 7–11. IEEE, 2009. [Pg.26]

[73] Stuart Schechter, Gabriel H Loh, Karin Strauss, and Doug Burger. Use ECP, not ECC, for hard failures in resistive memories. *ACM SIGARCH Computer Architecture News*, 38(3):141–152, 2010. [Pg.28], [Pg.41]

[74] Jue Wang, Xiangyu Dong, Yuan Xie, and Norman P Jouppi. i2 WAP: 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)*, pages 234–245. IEEE, 2013. [Pg.28], [Pg.98], [Pg.99], [Pg.126]

[75] Su-Kyung Yoon, Jitae Yun, Jung-Geun Kim, and Shin-Dug Kim. Self-adaptive filtering algorithm with PCM-based memory storage system. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(3):1–23, 2018. [Pg.31], [Pg.32]

[76] Dongsuk Shin, Hakbeom Jang, Kiseok Oh, and Jae W Lee. An energy-efficient DRAM cache architecture for mobile platforms with PCM-based main memory. *ACM Transactions on Embedded Computing Systems (TECS)*, 21(1):1–22, 2022. [Pg.31], [Pg.32]

[77] Zhe Wang, Shuchang Shan, Ting Cao, Junli Gu, Yi Xu, Shuai Mu, Yuan Xie, and Daniel A Jiménez. WADE: Writeback-aware dynamic cache management for NVM-based main memory system. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(4):1–21, 2013. [Pg.32], [Pg.34]

[78] Miao Zhou, Yu Du, Bruce Childers, Rami Melhem, and Daniel Mossé. Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems. *ACM Transactions on Architecture and Code Optimization (TACO)*, 8(4): 1–21, 2012. [Pg.32]

[79] Mohammad Bakhshalipour, Aydin Faraji, Seyed Armin Vakil Ghahani, Farid Samandi, Pejman Lotfi-Kamran, and Hamid Sarbazi-Azad. Reducing writebacks through in-cache displacement. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 24(2):1–21, 2019. [Pg.33], [Pg.34]

[80] Bahareh Pourshirazi, Majed Valad Beigi, Zhichun Zhu, and Gokhan Memik. Writeback-aware llc management for PCM-based main memory systems. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 24(2):1–19, 2019. [Pg.32], [Pg.33], [Pg.34]

[81] Hai Jin, Di Chen, Haikun Liu, Xiaofei Liao, Rentong Guo, and Yu Zhang. Miss penalty aware cache replacement for hybrid memory systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4669–4682, 2020. [Pg.32], [Pg.33], [Pg.34]

[82] Jie Xu, Dan Feng, Yu Hua, Wei Tong, Jingning Liu, Chunyan Li, and Zheng Li. An efficient PCM-based main memory system via exploiting fine-grained dirtiness of cachelines. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1616–1621. IEEE, 2018. [Pg.32], [Pg.33]

[83] Majid Jalili and Hamid Sarbazi-Azad. Captopril: Reducing the pressure of bit flips on hot locations in non-volatile main memories. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1116–1119. IEEE, 2016. [Pg.35]

[84] G David Forney. Coset codes. i. introduction and geometrical classification. *IEEE Transactions on Information Theory*, 34(5):1123–1151, 1988. [Pg.35]

[85] G David Forney. Coset codes. ii. binary lattices and related codes. *IEEE Transactions on Information Theory*, 34(5):1152–1187, 1988. [Pg.35]

[86] Seyed Mohammad Seyedzadeh, Rakan Maddah, Alex Jones, and Rami Melhem. Pres: Pseudo-random encoding scheme to increase the bit flip reduction in the memory. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015. [Pg.35]

[87] Suzhen Wu, Jiapeng Wu, Zhirong Shen, Zhihao Zhang, Zuocheng Wang, and Bo Mao. SimiEncode: A similarity-based encoding scheme to improve performance and lifetime of non-volatile main memory. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 220–227. IEEE, 2021. [Pg.35]

[88] Shihao Song, Anup Das, Onur Mutlu, and Nagarajan Kandasamy. Improving phase change memory performance with data content aware access. In *Proceedings of the 2020 ACM SIGPLAN International Symposium on Memory Management*, pages 30–47, 2020. [Pg.35]

[89] Alaa Alameldeen and David Wood. Frequent pattern compression: A significance-based compression scheme for l2 caches. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2004. [Pg.35], [Pg.36], [Pg.39], [Pg.94], [Pg.106], [Pg.108], [Pg.136], [Pg.140], [Pg.155]

[90] Gennady Pekhimenko, Vivek Seshadri, Onur Mutlu, Michael A Kozuch, Phillip B Gibbons, and Todd C Mowry. Base-delta-immediate compression: Practical data compression for on-chip caches. In *2012 21st international conference on parallel architectures and compilation techniques (PACT)*, pages 377–388. IEEE, 2012. [Pg.35], [Pg.36], [Pg.136], [Pg.140], [Pg.144], [Pg.155]

[91] David B Dgien, Poovaiah M Palangappa, Nathan A Hunter, Jiayin Li, and Kartik Mohanram. Compression architecture for bit-write reduction in non-volatile memory technologies. In *2014 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 51–56. IEEE, 2014. [Pg.36], [Pg.37]

[92] Seungcheol Baek, Hyung Gyu Lee, Chrysostomos Nicopoulos, and Jongman Kim. A dual-phase compression mechanism for hybrid DRAM/PCM main memory architectures. In *Proceedings of the Great lakes symposium on VLSI*, pages 345–350, 2012. [Pg.36]

[93] Yuncheng Guo, Yu Hua, and Pengfei Zuo. DFPC: A dynamic frequent pattern compression scheme in nvm-based main memory. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1622–1627. IEEE, 2018. [Pg.36], [Pg.37]

[94] Yuncheng Guo, Yu Hua, and Pengfei Zuo. A latency-optimized and energy-efficient write scheme in nvm-based main memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(1):62–74, 2018. [Pg.36]

[95] Zhangyu Chen, Yu Hua, Pengfei Zuo, Yuanyuan Sun, and Yuncheng Guo. Reducing bit writes in non-volatile main memory by similarity-aware compression. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. [Pg.36]

[96] Dan Feng, Jie Xu, Yu Hua, Wei Tong, Jingning Liu, Chunyan Li, and Yiran Chen. A low-overhead encoding scheme to extend the lifetime of nonvolatile memories. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10): 2516–2529, 2019. [Pg.36]

[97] Poovaiah M Palangappa and Kartik Mohanram. Flip-Mirror-Rotate: An architecture for bit-write reduction and wear leveling in non-volatile memories. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 221–224, 2015. [Pg.36], [Pg.37], [Pg.41], [Pg.108]

[98] Siddhartha Chhabra and Yan Solihin. i-NVMM: A secure non-volatile main memory system with incremental encryption. In *2011 38th Annual international symposium on computer architecture (ISCA)*, pages 177–188. IEEE, 2011. [Pg.37]

[99] Jingfei Kong and Huiyang Zhou. Improving privacy and lifetime of PCM-based main memory. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pages 333–342. IEEE, 2010. [Pg.37], [Pg.39]

[100] Pengfei Zuo, Yu Hua, Ming Zhao, Wen Zhou, and Yuncheng Guo. Improving the performance and endurance of encrypted non-volatile main memory through deduplicating writes. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 442–454. IEEE, 2018. [Pg.37], [Pg.38], [Pg.123], [Pg.144]

[101] Majid Jalili and Hamid Sarbazi-Azad. Endurance-aware security enhancement in non-volatile memories using compression and selective encryption. *IEEE Transactions on Computers*, 66(7):1132–1144, 2016. [Pg.37], [Pg.39], [Pg.134], [Pg.135]

[102] Poovaiah M Palangappa and Kartik Mohanram. CASTLE: Compression architecture for secure low latency, low energy, high endurance NVMs. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018. [Pg.39], [Pg.135]

[103] Wei Zhao, Dan Feng, Yu Hua, Wei Tong, Jingning Liu, Jie Xu, Chunyan Li, Gaoxiang Xu, and Yiran Chen. MORE 2: Morphable encryption and encoding for secure nvm. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2021. [Pg.37], [Pg.39]

[104] Mohammad Khavari Tavana, Yunsi Fei, and David Kaeli. Nacre*: Durable, secure and energy-efficient non-volatile memory utilizing data versioning. *IEEE Transactions on Emerging Topics in Computing*, 8(04):897–906, 2020. [Pg.37], [Pg.38], [Pg.39]

[105] Amro Awad, Pratyusa Manadhata, Stuart Haber, Yan Solihin, and William Horne. Silent shredder: Zero-cost shredding for secure non-volatile main memory controllers. *ACM SIGPLAN Notices*, 51(4):263–276, 2016. [Pg.37], [Pg.38]

[106] Stephen Longofono, Seyed Mohammad Seyedzadeh, and Alex K Jones. Virtual coset coding for encrypted non-volatile memories with multi-level cells. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 1128–1140. IEEE, 2022. [Pg.39]

[107] Dimin Niu, Qiaosha Zou, Cong Xu, and Yuan Xie. Low power multi-level-cell resistive memory design with incomplete data mapping. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 131–137. IEEE, 2013. [Pg.39]

[108] Moinuddin K Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *2009 42nd Annual IEEE/ACM international symposium on microarchitecture (MICRO)*, pages 14–23. IEEE, 2009. [Pg.40], [Pg.42]

[109] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S Lee. Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. *ACM SIGARCH computer architecture news*, 38(3): 383–394, 2010. [Pg.41]

[110] Jianming Huang, Yu Hua, Pengfei Zuo, Wen Zhou, and Fangting Huang. An efficient wear-level architecture using self-adaptive wear leveling. In *49th International Conference on Parallel Processing-ICPP*, pages 1–11, 2020. [Pg.41], [Pg.42]

[111] Marjan Asadinia, Majid Jalili, and Hamid Sarbazi-Azad. BLESS: A simple and efficient scheme for prolonging PCM lifetime. In *Proceedings of the 53rd Annual Design Automation Conference*, pages 1–6, 2016. [Pg.41], [Pg.42], [Pg.99]

[112] Xianzhang Chen, Zhuge Qingfeng, Qiang Sun, Edwin H-M Sha, Shouzhen Gu, Chaoshu Yang, and Chun Jason Xue. A wear-leveling-aware fine-grained allocator for non-volatile memory. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019. [Pg.41]

[113] Shivam Swami and Kartik Mohanram. ACME: Advanced counter mode encryption for secure non-volatile memories. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018. [Pg.42], [Pg.111], [Pg.121]

[114] Shivam Swami and Kartik Mohanram. COVERT: Counter OVErflow ReducTion for efficient encryption of non-volatlle memories. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 906–909. IEEE, 2017. [Pg.42]

[115] Fangting Huang, Dan Feng, Yu Hua, and Wen Zhou. A wear-leveling-aware counter mode for data encryption in non-volatile memories. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 910–913. IEEE, 2017. [Pg.42]

[116] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 72–81, Oct 2008. [Pg.45], [Pg.59], [Pg.68], [Pg.82], [Pg.93], [Pg.123], [Pg.140], [Pg.148], [Pg.165]

[117] Di Chen, Hai Jin, Xiaofei Liao, Haikun Liu, Rentong Guo, and Dong Liu. MALRU: Miss-penalty aware LRU-based cache replacement for hybrid memory systems. In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '17, pages 1086–1091, 3001 Leuven, Belgium, Belgium, 2017. European Design and Automation Association. URL `http://dl.acm.org/citation.cfm?id=3130379.3130637`. [Pg.46]

[118] G. Keramidas, P. Petoumenos, and S. Kaxiras. Cache replacement based on reuse-distance prediction. In *2007 25th International Conference on Computer Design*, pages 245–250, Oct 2007. doi: 10.1109/ICCD.2007.4601909. [Pg.49]

[119] J. Albericio, P. Ibez, V. Vials, and J. M. Llabera. The reuse cache: Downsizing the shared last-level cache. In *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 310–321, Dec 2013. [Pg.49]

[120] G. Kurian, S. Devadas, and O. Khan. Locality-aware data replication in the last-level cache. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12, Feb 2014. doi: 10.1109/HPCA.2014.6835921. [Pg.46]

[121] Zhigang Hu, Stefanos Kaxiras, and Margaret Martonosi. Timekeeping in the memory system: Predicting and optimizing memory behavior. *SIGARCH Comput. Archit. News*, 30(2):209–220, May 2002. ISSN 0163-5964. [Pg.47]

[122] Jamison D. Collins and Dean M. Tullsen. Runtime identification of cache conflict misses: The adaptive miss buffer. *ACM Trans. Comput. Syst.*, 19(4):413–439, November 2001. ISSN 0734-2071.

[123] A. Basu, N. Kirman, M. Kirman, M. Chaudhuri, and J. Martinez. Scavenger: A new last level cache architecture with global block priority. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, pages 421–432, Dec 2007. [Pg.47]

[124] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The Gem5 Simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011. ISSN 0163-5964. doi: 10.1145/2024716.2024718. URL `http://doi.acm.org/10.1145/2024716.2024718`. [Pg.47], [Pg.59], [Pg.93], [Pg.123], [Pg.139], [Pg.148], [Pg.159]

[125] M. Poremba and Y. Xie. NVMain: An architectural-level main memory simulator for emerging non-volatile memories. In *2012 IEEE Computer Society Annual Symposium on VLSI*, pages 392–397, Aug 2012. [Pg.47], [Pg.59], [Pg.93], [Pg.123], [Pg.139], [Pg.148]

[126] Aamer Jaleel, Kevin B. Theobald, Simon C. Steely, Jr., and Joel Emer. High performance cache replacement using re-reference interval prediction (RRIP). *SIGARCH Comput. Archit. News*, 38(3):60–71, June 2010. ISSN 0163-5964. doi: 10.1145/1816038.1815971. URL `http://doi.acm.org/10.1145/1816038.1815971`. [Pg.48]

[127] Norman P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. *SIGARCH Comput. Archit. News*, 18(2SI):364–373, May 1990. ISSN 0163-5964. [Pg.49]

[128] Moinuddin K. Qureshi and Yale N. Patt. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 423–432, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2732-9. doi: 10.1109/MICRO.2006.49. URL `https://doi.org/10.1109/MICRO.2006.49`. [Pg.55]

[129] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, July 2012. ISSN 0278-0070. [Pg.60], [Pg.73], [Pg.162]

[130] S. J. E. Wilton and N. P. Jouppi. CACTI: an enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, May 1996. ISSN 0018-9200. [Pg.60], [Pg.73], [Pg.123]

[131] Christian Bienia, Sanjeev Kumar, and Kai Li. PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors. In *2008 IEEE International Symposium on Workload Characterization*, pages 47–56. IEEE, 2008. [Pg.68]

[132] M. Bhadauria, V. M. Weaver, and S. A. McKee. Understanding PARSEC performance on contemporary CMPs. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*, pages 98–107, Oct 2009. [Pg.68]

[133] John L Henning. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, 2006. [Pg.82], [Pg.93], [Pg.123], [Pg.140], [Pg.148], [Pg.165]

[134] A N. Jacobvitz et al. Coset coding to extend the lifetime of memory. In *HPCA*, pages 222–233, Feb 2013. [Pg.85], [Pg.86], [Pg.94]

[135] Mahdi Nazm Bojnordi and Engin Ipek. PARDIS: A programmable memory controller for the DDRx interfacing standards. *ACM SIGARCH Computer Architecture News*, 40(3):13–24, 2012. [Pg.106], [Pg.144]

[136] Jinho Lee, Jongwook Chung, Jung Ho Ahn, and Kiyoung Choi. Excavating the hidden parallelism inside dram architectures with buffered compares. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(6):1793–1806, 2017. doi: 10.1109/TVLSI.2017.2655722. [Pg.106], [Pg.144]

[137] Mohammad Alshboul, Hussein Elnawawy, Reem Elkhouly, Keiji Kimura, James Tuck, and Yan Solihin. Efficient checkpointing with recompute scheme for non-volatile main

memory. *ACM Transactions on Architecture and Code Optimization (TACO)*, 16(2): 1–27, 2019. [Pg.110]

[138] Hyojong Kim, Hongyeol Lim, Dilan Manatunga, Hyesoon Kim, and Gi-Ho Park. Accelerating application start-up with nonvolatile memory in android systems. *IEEE Micro*, 35(1):15–25, 2015. [Pg.110]

[139] Jun Yang and Rajiv Gupta. Frequent value locality and its applications. *ACM Transactions on Embedded Computing Systems (TECS)*, 1(1):79–105, 2002. [Pg.118]

[140] Jun Yang, Youtao Zhang, and Rajiv Gupta. Frequent value compression in data caches. In *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, pages 258–265, 2000. [Pg.118]

[141] Georgios Keramidas, Konstantinos Aisopos, and Stefanos Kaxiras. Dynamic dictionary-based data compression for level-1 caches. In *Architecture of Computing Systems-ARCS 2006: 19th International Conference, Frankfurt/Main, Germany, March 13-16, 2006. Proceedings 19*, pages 114–129. Springer, 2006. [Pg.118]

[142] Jun Yang, Rajiv Gupta, and Chuanjun Zhang. Frequent value encoding for low power data buses. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 9(3):354–384, 2004. [Pg.118]

[143] Weihua Zhang, Jiaxin Li, Yi Li, and Haibo Chen. Multilevel phase analysis. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(2):1–29, 2015. [Pg.121]

[144] Canturk Isci, Alper Buyuktosunoglu, and Margaret Martonosi. Long-term workload phases: Duration predictions and applications to DVFs. *Ieee Micro*, 25(5):39–51, 2005. [Pg.121]

[145] Weidong Shi, H-h S Lee, Mrinmoy Ghosh, Chenghuai Lu, and Alexandra Boldyreva. High efficiency counter mode security architecture via prediction and precomputation. In *32nd International Symposium on Computer Architecture (ISCA'05)*, pages 14–24. IEEE, 2005. [Pg.123], [Pg.148]

[146] Sanu Mathew, Sudhir Satpathy, Vikram Suresh, Mark Anders, Himanshu Kaul, Amit Agarwal, Steven Hsu, Gregory Chen, and Ram Krishnamurthy. 340 mv–1.1 v, 289 gbps/w, 2090-gate nanoaes hardware accelerator with area-optimized encrypt/decrypt gf (2 4) 2 polynomials in 22 nm tri-gate cmos. *IEEE Journal of Solid-State Circuits*, 50(4):1048–1058, 2015. [Pg.132]

[147] `https://ark.intel.com/content/www/us/en/ark.html`. [Pg.132]

[148] Arijit Nath and Hemangee K Kapoor. WELCOMF: wear leveling assisted compression using frequent words in non-volatile main memories. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 157–162, 2020. [Pg.136], [Pg.140], [Pg.145]

[149] Arijit Nath and Hemangee K Kapoor. SWEL-COFAE: Wear leveling and adaptive encoding assisted compression of frequent words in non-volatile main memories. *IEEE Transactions on Computers*, 2021. [Pg.136]

[150] Shyamkumar Thoziyoor, N Muralimanohar, J Ahn, and N Jouppi. Cacti 6.5. *hpl. hp. com*, 2009. [Pg.162]

[151] ITRS 2.0 Home Page [online]. Available : http://www.itrs2.net/". [Pg.163]

Department of Comp Science and Engineering
Indian Institute of Technology Guwahati
Guwahati 781039, India