2017

Energy-Aware Data Movement In Non-Volatile Memory Hierarchies

Navid Khoshavi Najafabadi
University of Central Florida

ENERGY-AWARE DATA MOVEMENT IN NON-VOLATILE MEMORY HIERARCHIES

by

NAVID KHOSHAVI NAJAFABADI
M.S. University of Central Florida, 2016
M.S. Amirkabir University of Technology, 2012

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2017

Major Professor: Ronald F. DeMara

ABSTRACT

While technology scaling enables increased density for memory cells, the intrinsic high leakage power of conventional CMOS technology and the demand for reduced energy consumption inspires the use of emerging technology alternatives such as eDRAM and Non-Volatile Memory (NVM) including STT-MRAM, PCM, and RRAM. The utilization of emerging technology in Last Level Cache (LLC) designs which occupies a significant fraction of total die area in Chip Multi Processors (CMPs) introduces new dimensions of vulnerability, energy consumption, and performance delivery. To be specific, a part of this research focuses on eDRAM Bit Upset Vulnerability Factor (BUVF) to assess vulnerable portion of the eDRAM refresh cycle where the critical charge varies depending on the write voltage, storage and bit-line capacitance. This dissertation broaden the study on vulnerability assessment of LLC through investigating the impact of Process Variations (PV) on narrow resistive sensing margins in high-density NVM arrays, including on-chip cache and primary memory. Large-latency and power-hungry Sense Amplifiers (SAs) have been adapted to combat PV in the past. Herein, a novel approach is proposed to leverage the PV in NVM arrays using *Self-Organized Sub-bank (SOS)* design. SOS engages the preferred SA alternative based on the intrinsic as-built behavior of the resistive sensing timing margin to reduce the latency and power consumption while maintaining acceptable access time.

On the other hand, this dissertation investigates a novel technique to prioritize the service to 1) Extensive Read Reused Accessed blocks of the LLC that are silently dropped from higher levels of cache, and 2) the portion of the working set that may exhibit distant re-reference interval in L2. In particular, we develop a lightweight Multi-level Access History Profiler to efficiently identify ERRA blocks through aggregating the LLC block addresses tagged with identical Most Significant Bits into a single entry. Experimental results indicate that the proposed technique can reduce the L2 read miss ratio by 51.7% on average across PARSEC and SPEC2006 workloads.

In addition, this dissertation will broaden and apply advancements in theories of subspace recovery to pioneer computationally-aware in-situ operand reconstruction via the novel Logic In Interconnect (LI2) scheme. LI2 will be developed, validated, and refined both theoretically and experimentally to realize a radically different approach to post-Moore's Law computing by leveraging low-rank matrices features offering data reconstruction instead of fetching data from main memory to reduce energy/latency cost per data movement. We propose LI2 enhancement to attain high performance delivery in the post-Moore's Law era through equipping the contemporary micro-architecture design with a customized memory controller which orchestrates the memory request for fetching low-rank matrices to customized Fine Grain Reconfigurable Accelerator (FGRA) for reconstruction while the other memory requests are serviced as before. The goal of LI2 is to conquer the high latency/energy required to traverse main memory arrays in the case of Last Level Cache (LLC) miss, by using in-situ construction of the requested data dealing with low-rank matrices. Thus, LI2 exchanges a high volume of data transfers with a novel lightweight reconstruction method under specific conditions using a cross-layer hardware/algorithm approach.

I dedicate my dissertation work to my family and friends. A special feeling of gratitude to my wife, Sahar, whose affection, love, encouragement make me able to get such success and honor. A special feeling of thankful to my parents whose unlimited support either financially or intellectually made this work possible. I dedicate this work and special thanks to not only my advisor, but also one of my friends for sharing his knowledge about how I can become a successful researcher.

ACKNOWLEDGMENTS

I owe my gratitude to many people who made this thesis possible even only my name will be appeared on the cover of this thesis. I have been amazingly fortunate to work under supervision of Dr. DeMara who patiently supported me even in tough situations and continuously motivated me to become a successful student. His insightful comments and immense knowledge helped me to shape my ideas at different stage of my research which leads to publishing several papers in the well-known conferences and journals. I would like to extend special thanks to Dr. Jun Wand, Dr. Mingjie Lin, Dr. Cliff Zou, and Dr. Damian Dechev for serving in my dissertation committee.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION AND MOTIVATION

The CMOS technology advancements over past decades has introduced more accurate and cost-efficient techniques for manufacturing the electronic devices. The demand for delivering greater performance in a smaller chip area has motivated the International Technology Roadmap for Semiconductors (ITRS) community to significantly investigate the challenges of entering the nanometer scale CMOS technology era [24]. Even though the reduction of transistors gate length size in the early stage of CMOS microprocessor design process has promised the continued process of doubling the quantity of transistors in a single die with Moors Law [137], the power wall challenge [146] has enforced CMOS microprocessor designers to shift towards develop of multi-core processors. This design paradigm has made the researchers to confront new design and implementation obstacles where we find a way to turn them into opportunities for developing cost-effective and energy efficient high performance system designs. In the remaining of this chapter, we will discuss about these challenges and opportunities.

1.1 Soft Error Impacts on eDRAM-based LLC Design

Large Last Level Cache (LLC) consumes a large fraction of the die in modern CMPs. High bandwidth demands of memory-intensive applications utilizing large working sets require larger LLC to (1) facilitate the residency of the working set to remain in cache, (2) prevent data replacement due to continual communication between cores. As the technology goes down toward sub-micron feature size, more cache lines can fit on the same LLC area which increases the cache capacity resulting in delivery of reduced miss rate to all cores. However, this approach is coupled with vulnerability increase of LLC against Single Event Upset (SEU) and Multi-Bit Upsets (MBUs).

Soft errors induced by energetic particles penetrate the silicon substrate and generate electron-hole pairs along their tracks. If the generated electrons collected into cell junction is larger than critical charge (Q_C), it can flip the cell state and generate soft error in memory cell [153]. Even though the probability of Soft Error Rate (SER) per bit is small, the large density of LLC unacceptably increases the SER per chip. In particular, given roughly 50% of chip is occupied by cache memory structure, high-dense large-scale LLC counterpart, becomes highly susceptible to soft errors [158]. Moreover, the high potential of residing a data block in LLC for millions of cycles between two consecutive accesses has significantly increased the vulnerability of LLC cache blocks to soft errors [111].

Recently, large embedded Dynamic Random Access Memory (eDRAM) cache has been introduced as the LLC or L3 cache between L2 SRAM cache and main memory to alleviate further the core-memory speed gap [111][103]. The employment of on-chip L3 eDRAM offers higher cache capacity compared to SRAM [111][110][170] and provides faster on-chip communication through on-chip high bandwidth [17][37]. The eDRAM requires periodic refresh operation less than $retention\ time$ to prevent data loss due to the leakage of capacitors over time. The eDRAM cell's retention time is defined as the amount of time that eDRAM cell can preserve data before data loss occurs [145][29]. As the feature size continues to decrease, the eDRAM cell size reduction imposes serious problems to eDRAM design because it not only decreases the retention time of eDRAM cell, but also leads to reduced reliability in terms of increasing the susceptibility of eDRAM cell to the soft errors [16][57][4][47].

In order to preserve data integrity in eDRAM, the charge in each capacitor must be refreshed periodically. This means the data stored in eDRAM is required to be read out and written back into cell by each refresh operation. This new lifetime sequence needs to be investigated to accurately reveal how different lifetime sequences of cache data contribute to vulnerability. Thus, despite the fact that the most previous works studied the trade-offs between performance, energy consumption and achieved cache reliability, more systematic study of cache vulnerability is still needed to present a detailed lifetime model for data arrays in eDRAM LLC. Given these challenges, we propose a more accurate and application-relevant lifetime model in Chapter 3 while taking the refresh operation scheme for LLC into consideration. The proposed lifetime model distinguishes among six lifetime sequences for each data item located in eDRAM LLC according to the previous activity and current one. These sequences are further categorized into two groups, $vulnerable$ and $nonvulnerable$ sequences. A vulnerable sequence is characterized by the fact that any error that is encountered during this sequence has the potential to be either consumed by the CPU or committed to memory by a write. However, if the exposure occurs during nonvulnerable sequence, no program failure will result.

1.2 Process Variation Impact on NVM-based LLC Design

Recently, the emerging Non-Volatile Memory (NVM) technologies such as Resistive RAM (RRAM), Phase Change Memory (PCM) and Spin Transfer Torque Magnetic RAM (STT-MRAM) have received significant attention as promising approaches to overcome the CMOS-based memory storage challenges such as volatility and high static power consumption. Even though the utilization of NVM technology in the overarching system can significantly reduce the leakage power issue, the Process Variation (PV) effect has still remained as a limiting factor for NVM applicability and scalability. In particular, the PV in Magnetic Tunnel Junction (MTJ) devices utilized in STT-MRAM organization, manifests itself as variation in MgO thickness and MTJ geometry which in turn results in deviation of MTJ resistance [159]. Furthermore, the threshold voltage, V_{th}, and gate length, L_{eff}, of CMOS access transistors in STT-MRAM organization exhibit delay and driving current variations under PV, which negatively impacts the performance consistency of memory operation [43].

As the results of PV effect on both conventional and emerging semiconductor technologies, the difference between the sensed bit-line voltage and the reference voltage which is referred as *sense margin* can severely fluctuate, resulting in possible false detection scenario and increased bit error rate [184]. This issue has increased the demand for designing advanced low-power and reliable sensing circuits which can be integrated into PV-resilient system architectures while providing required sensing margin. To provide reliable sensing operation while taking the energy budget into consideration, we propose a circuit-architecture cross-layer solution suitable for multi-core processors as well as IoT devices.

Our proposed technique, referred to as *Self-Organized Sub-bank (SOS)*, partitions STT-MRAM data arrays into several sub-banks to directly access requested data while introducing individualized sensing resolution. Sub-banks are evaluated and tagged during an initial *Power-On Self-Test (POST)* phase to identify the preferred SA for that particular sub-bank. Hence, SOS reduces the risk of contaminating the application's data structure by fault propagation.

1.3 The Impact of Data Movement on Overall Performance and Energy Consumption

Apart from reliability concerns for correctly sensing data from LLC, the increasing bandwidth demand of current memory-intensive applications incurs significant data movement that negatively impacts off-chip bandwidth, on-chip memory access latency, and energy consumption [19] [102] [30] [6]. To reduce data transfer between on-chip and off-chip memory components, commercial multi-core systems utilize multi-level cache methodology [20] [93] [74] whereby fast, low-capacity, and high leakage power SRAM arrays are employed in the upper-levels of cache, i.e. L1 and L2, while large, low leakage power and high refresh demand eDRAM is placed in LLC. The employment of relatively spacious SRAM arrays as L2 cache design in the middle of cache hierarchy results in two major challenges: 1) *leakage*: the high leakage power characteristic of

SRAM cells results in excessive power budget [79], and 2) *area*: capacity constraints induced by the SRAM cell footprint prevent favorable residency of the working set to reside close to the active core [75] [72]. The negative effect of aforementioned issues is exacerbated with the high dynamic power dissipation incurred to drive on-chip interconnects while exchanging data [18][165].

On the other hand, the cache memory shortage to maintain continuously expanding working sets close to the core causes the performance degradation or increased miss ratio. This degradation is exacerbated by introducing more cache levels with larger LLC capacity. For instance, the transfer latency for a cache block from L2 to L1 in hyper-threaded Pentium IV is 18 cycles, while this latency goes up to 360 cycles for data movement between main memory and L2 [164]. Furthermore, the existing cache replacement policies are ineffective to deal with continuously growing large memory-intensive workloads that typically are greater than available cache size [138]. In particular, a fraction of working set that exhibit distant re-reference interval may repeatedly be evicted and be brought back from/to L2 prior to contributing sufficient hits to L2. Thus, the processor may stall for a long time to access the data stored in the cache blocks in LLC if there are no independent instructions to execute.

While other recent works have made significant advancement to optimize eDRAM LLC (L3 in this work) [112, 11], in Chapter 5, we introduce new insights regarding working set behavior to optimize L2 cache using a heterogeneous STT-MRAM. Various hybrid cache designs have been presented in the past to maintain the read-intensive cache blocks in the high-retention region while write-intensive blocks are allocated to low-retention regions of cache [161, 81]. The same strategy is adopted in our design for prioritizing the service to 1) Extensive Read Reused Access (ERRA) blocks that remain unchanged during their residency in LLC, and 2) frequently reused blocks which may exhibit distant re-reference intervals in L2. An efficient block allocation/replacement policy is introduced to maximize the throughput of the adopted hybrid cache design. To minimize the overhead of finding and labeling the potential blocks that must be prioritized to copy into high-retention regions of the adopted hybrid L2 cache, we propose a novel Multi-level Access History Profiler (MAHP) inspired by observing the high spatial locality at page-level granularity among different class of workloads. To be specific, we observed that more than 50% of read accesses are from one of the three pages with the most accesses in the selected workloads, indicating that the majority of cache block accesses are from the same physical page. MAHP aggregates the LLC block addresses tagged with identical Most Significant Bits (MSBs) into a single entry while their remaining Least Significant Bits (LSBs) of the addresses are stored into distinct LSB-address entries.

1.4 The Impact of Big Data Processing on Memory Throughput

The increasing demand for Big Data processing often requires the parallel execution of vast number of servers to employ multifaceted analytical methods for extracting meaningful value from data. Nowadays, around 80 percent of the large data-sets are unstructured data exhibiting no specified format which makes the datum access, arrangement, and processing extremely formidable [70]. To be specific, the large volume of unstructured data are analyzed with data structures that often incur random access patterns, which in turn cause cache underutilization, high memory access conflict, increased power dissipation for driving on-chip interconnects and activating off-chip main memory components (row/column address strobe, bit line pre-charge, sensing operation), and reduced memory-level parallelism. Therefore, research into addressing the computational challenges associated with unstructured data has significant potential to transform scalable parallelism using a cross-layer computation fashion from the application layer down to the micro-architecture.

Over the past few decades, the focus of researchers were mostly on speeding up the computational capabilities in traditional computing-centric model which is based on moving large volume of data from/to memory storage to/from processing nodes for execution/store. To maximize the efficacy of computing-centric model, many fine/coarse -grain parallelism paradigms from software approach, i.e. instruction/data/thread/transaction -level parallelism [180, 84, 22, 39, 178, 64, 144, 38], to micro-architecture domain, i.e. pipelining [172, 77, 113], superscalar [15, 152], VLIW [109], Single Instruction Multiple Data (SIMD) instructions [118, 122], vector processors [125], Graphic Processor Unit (GPUs) [128, 53], chip-multiprocessors [35, 183], clusters [23, 56, 34], have been proposed in the past as illustrated in Figure 1.1, that facilitate the concurrent execution of numerous fine-grained threads on multiple functional units for arithmetic execution and logical calculation. For example, recently Nvidia company has released Tesla P100(SXM2) GPU accelerator that consumes about 56pJ per double-precision floating point operation derived from max power consumption of 300W for 5.3 Tflops operations [2]. Even though this achievement is approximately close to the anticipated 20pJ energy cost for targeted power envelop which was projected on 2008 for contemporary supercomputers [96], the simultaneous activation of multiple functional units has faced the power wall challenge which is referred as the dark silicon effect. This phenomenon stems from the failure of Dennard scaling where the aggressive CMOS technology scaling is no longer in line with voltage scaling, resulting in sharp increase of power density [162].

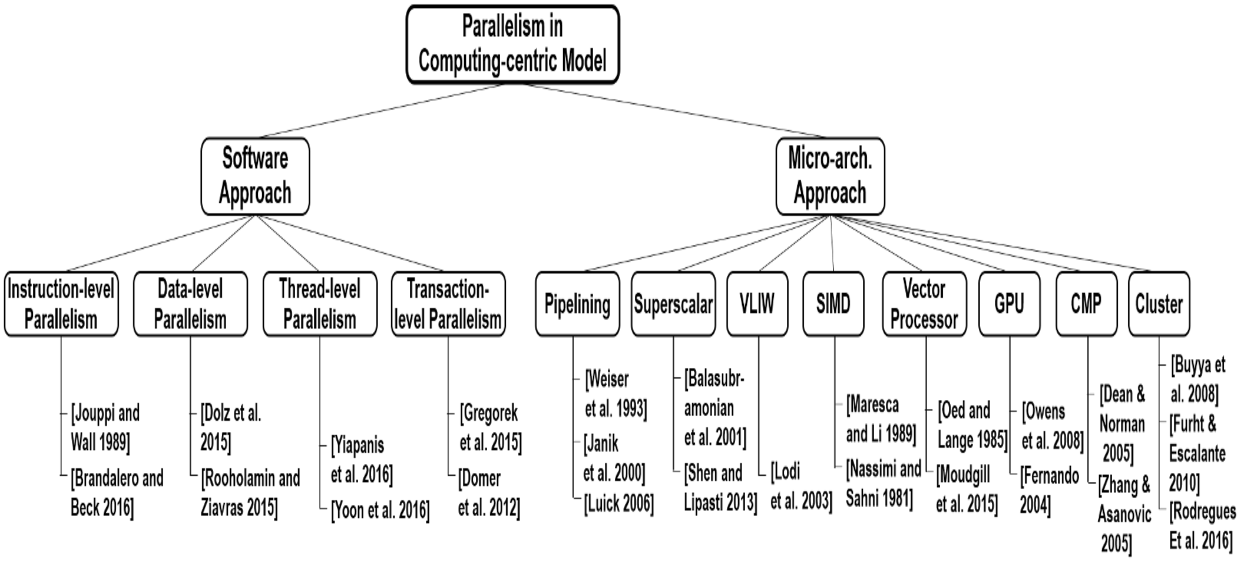


Figure 1.1: Taxonomy of parallelism techniques in the traditional computing-centric model, which illustrates opportunities to redesign the location, plurality, and role of memory along with its interfaces.

This encourages the computer architecture designers to shift the paradigm of computation from computing-centric model towards data-centric architecture for further parallelism and minimizing the energy cost per computation operation. The processing nodes in data-centric model are arranged to be placed close to the location of data, resulting in remarkable reduction of the cost for data movement [14]. These achievements have been coupled with the advancements in technology and manufacturing for 3D die stacking, and custom memory interfaces such as High Bandwidth Memory (HBM) and Hybrid Memory Cube (HMC) which have enabled the efficient implementation of data-centric models. In particular, direct data Processing-In-Memory (PIM) [44, 62, 126, 65, 66, 7, 9, 42, 88, 116], near memory processing [49, 50, 10, 171, 48, 182, 188, 149], and processing in memory controller [46, 179] have received significant attention for implementing data-centric architecture. However, means for continuing scalable parallelism especially in scientific applications will be through the development of a software-architecture cross-layer approach which is orthogonal to the recent data-centric model based techniques. The proposed method must reduce further the on/off -chip interferences for accessing shared resources while eliminating unnecessary multiple round-trip to main memory for fetch and update of the data, which in turn results in the reduced orchestration workflow and data movement.

To attain this urgent goal, we target a category of matrices that are widely observed in Big Data processing, i.e. dimension reduction, signal processing, compression, clustering, regression, and classification, called the *low rank matrix*, as a high-payoff strategy to reduce data movement across the entire system stack. In particular, the low rank matrices exhibit specific characteristics that enable computationally-efficient reconstruction of the required column using two decomposed matrices which have significantly fewer elements than the original matrix, at the cost of a slight storage overhead. We argue that these two matrices are updated rarely in our target applications, which means that the overhead for extracting them from original matrix is correspondingly small.

CHAPTER 2: RELATED WORK

2.1 Soft Error Vulnerability Analysis in Memory Storage Organization

Asadi et al. [12] proposed algorithms of vulnerability computation for both L1 and L2 caches. The authors indicated the vulnerability breakdown of data, tag-addresses, and status bits. Tag error is analyzed in detail and classified into three categories. Whereas Asadi′s research is forced on L1 and L2 private cache, Maghsoudloo et al. [114] analyzed the influence of coherence protocol on the susceptibility of shared LLC. Two prediction schemes are proposed to provide correction ability for dirty data.

The authors of [168] conducted a study to assess the reliability behavior of cache memories in order to provide insight into the cache design for manufacturing highly efficient reliable on-chip cache. Meanwhile, an analytical framework is a proposed by Suh et al. [158] to measure the failure rate in L2 SRAM cache under any soft error protection scheme. However, these two works are not specifically designed for eDRAM.

In terms of eDRAM and DRAM reliability analysis work, Fang et al. [47] developed an efficient method to predict scaling trends for both neutron- and alpha- soft error rate. However, our scheme is suitable for a wide range of soft errors. Shin [153] proposed a unified model for alpha-particle-induced charge collection. For soft error concerns outside of cache broader treatments are presented in [115][58].

2.2 Exploration of Hybrid Cache Design

Hybrid cache design approaches have received significant attention over past years. These techniques leverage the benefits of utilized technologies to maximize the performance and minimize the energy consumption. In [166], both SRAM and eDRAM technologies are employed in the second-level cache to offer area savings and reduced energy consumption while the performance is increased by 5.9 percent on average. The proposed LLC design leverages the fast SRAM device to store the most likely referenced blocks in the future while the high-dense and low-leakage eDRAM is utilized to reduce the overall energy consumption [166]. In other words, the larger SRAM banks can provide the higher performance while the deployment of larger area-efficient eDRAM requires lower energy for operation. Thus, the ratio of utilizing these technologies in the LLC has been optimized to provide the sufficient performance and reduce energy consumption. However, this design does not explicitly consider the critical load behavior.

The proposed work in [73] leverages the asymmetric write power associated with storing ’ones’ and ’zeros’ to place data blocks having majority ’zero’ data into STT-RAM while the remained data blocks are stored in the SRAM. The less energy is required to write ’zero’ in STT-RAM compared to writing ’one’. This asymmetric write power of the STT-RAM motivated the authors to place data blocks having majority ’zero’ data into STT-RAM while the remained data blocks are stored in the SRAM. Since the data block is updated several times during its lifetime, it is possible that the majority of data changes over time. Accordingly, a two-bit counter has been considered for each cache line to track the status of the majority. The migration policy swap the data blocks according to the pace of majority-data changes. In other word, it does not immediately move the data by observing the changes in the majority-data to avoid write overhead associated with the unnecessary migration. Thus, the migration policy transfers the data between two partitions in favor of reducing the write energy overhead. This design limits the performance of multi-core chip to the efficiency of the module which determines the majority of the data. Namely, each data block should be given to the dispatcher module before placement in the LLC which incurs significant overhead for memory-intensive applications imposing significant pressure on the placement policy.

The proposed hybrid LLC design proposed in [169] utilizes an intelligent adaptive data block placement by taking each cache line future access pattern into consideration. The write accesses are categorized into three main classes: prefetch-write, core-write and demand-write. Around 26% of all prefetch blocks are not accessed by the core after initial prefetch [169]. Furthermore, the data blocks moved to the LLC due to cache burst phenomenon are not accessed again until the eviction occurs [169]. The data blocks with the aforementioned characteristics are considered to be placed in the SRAM. On the other hand, the data blocks which experience long interval between consecutive reads are placed into the STT-RAM to benefit from the low leakage power cost of long-residency offered by non-volatile devices [169]. This design may incur prediction design overhead which incurs extra energy consumption for tracking the access pattern to each cache line and making the decision for placement.

In [161], Low-Retention (LR) and High-Retention (HR) STT-RAM arrays are utilized simultaneously to balance the performance versus the energy consumption. To accomplish this, the retention time of the STT-RAM is relaxed for LR architecture to improve the write operation speed. On the other hand, the HR cache offers the long-term residency of data block while incurring very small leakage power. Regarding to the features that each cache design offers, the proposed method manages the write-intensive cache blocks to be placed into the LR cache in favor of performance, while the read-intensive cache blocks are kept in the HR arrays to meet the required power budget limits. Furthermore, the migration policy is devised to transfer the data blocks to the proper cache design by continuously monitoring the access pattern to each cache line. Since the lifespan of the cache lines in the L1 arrays is limited to the retention time which is deliberated at the design time, a refresh mechanism is designed to periodically refresh cache lines for preventing data loss. RRAP utilizes the same refresh approach to maintain the stability of data in LRSC design.

In [147], the similar idea is utilized in the LLC design for GPU. The small-sized LR cache is employed to keep the write-intensive data blocks by considering the fact that the re-write interval time of blocks is typically lower than $100\mu s$. Again, the large HR arrays is considered to maintain the less-frequently written data blocks [147]. The proposed technique uses a write threshold on HR to determine whether to keep the data in HR or move it to the LR. To achieve this goal, the correlation between the threshold value and the performance has been accurately analyzed, and the optimum value has been selected as the threshold value. In addition, different degrees of associativity are considered for LR and HR designs to maximize the write utilization in LR and to improve the data migration policy. The taxonomy of the discussed techniques are presented in Fig. 2.1 and the contribution of our technique which will be presented in Chapter 5, compared to other techniques is highlighted in Table 2.1.

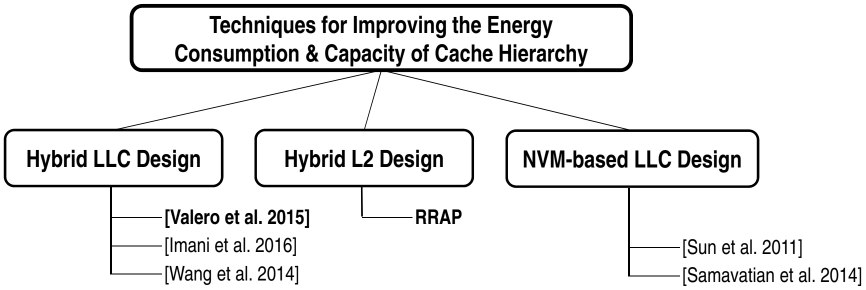


Figure 2.1: Taxonomy of techniques utilizing the emergent technologies for improving the performance and energy consumption. Approaches using eDRAM highlighted in bold face font.

Besides proposing hybrid cache designs to enhance performance and energy consumption, the bypass methods [63, 120, 141, 95] have received significant attention for their efficiency to improve the temporal locality. Gonzalez et al. [63] proposed a data cache organization to improve both spatial and temporal locality through *locality prediction table*. Essentially, the history of recently referenced instructions is considered to bypass branches that can cause cache pollution. McFarling [120] proposed to dynamically exclude cache lines that contribute to the conflict misses in a direct-mapped cache. Rathi et al. [141] proposed to bypass STT-RAM based LLC while hooking up directly upper level caches to the memory if the LLC is susceptible to Denial of Service (DoS) or any other attacks. Kim et al. [95] proposed a bypass method for inclusive NVM-based LLC to reduce the write operations overhead which in turn results in the improved performance and efficient energy consumption. However, both replacement policy and the available cache capacity determine the efficiency of the aforementioned techniques. For example, suppose the working set cannot fit into L2 due to the insufficient cache capacity. If the access sequences of the running application to the L2 blocks discard a portion of resided blocks whose will be accessed at the end of reference pattern, most of the newly inserted blocks will be dead before contributing sufficient hit to L2. This scenario may also occur in the applications that exhibit the burst reference pattern to non-temporal blocks [76]. Thus, even though bypass methods mitigate the poor locality in most of scenarios, in this case maintaining a highly read accessed fraction of the working set in the upper level caches for a long-enough period aids to improve the temporal locality.

Table 2.1: Comparison of hybrid cache design techniques.

Approach	**multi-retention STT-RAM employment**	**Response time to write operation**	**Energy consumption resolution**	**Service to critical loads**	**Cache configuration**	**Technique's overhead**
Sun et al.[161]	yes	fast (volatile STT-RAM)	low-leakage STT-RAM & data migration	no	N/A	periodic refresh+ exhaustive data migration
Valero et al.[166]	no	fast (SRAM banks)	Hybrid SRAM & eDRAM banks design	no	N/A	periodic refresh+ swap between SRAM & eDRAM
samavatian et al.[147]	yes	fast (volatile STT-RAM)	low-leakage STT-RAM & data migration	no	N/A	periodic refresh+ data migration
Jog et al.[79]	yes	fast (volatile STT-RAM)	low-leakage STT-RAM & buffering-based refresh scheme	no	N/A	write back dirty blocks+ buffering
RRAP (approach in Chapter 5)	yes	fast (volatile STT-RAM)	low-leakage STT-RAM & data movement reduction	yes	RRAPclusive	periodic refresh

2.3 In-Memory Data Management to Minimize Data Movement

The continuous demand for accelerating the analysis and process of Big Data has shifted the paradigm of conventional computing-centric model towards data-centric architecture. This transition is driven by the aggressive CMOS technology scaling trends and advancements in fabrication process which enable the integration of billions of transistors into a small-scale die area to elevate the performance. These achievements are coupled with the advent of many-core processors that provides numerous number of computation nodes to schedule and perform multitasking workloads. The employment of ultra-fast many-core processors in the modern computers has significantly accelerated the arithmetic execution and calculation of logic which were bottleneck of computation in the past. Thus, the nature of computing-centric model which is based on moving large volume of data from/to memory storage to/from processing nodes for execution/restore has become ineffective. In contrast, the data-centric model has received significant attention in which the processing nodes are arranged to be placed close to the location of data, resulting in remarkable reduction of the cost for data movement [14].

Apart from the advancements in designing high performance many-core processors, the progress of tightly stacking logic and memory elements in a package has enabled the researchers to redesign the conventional Processing-In-Memory (PIM) concept for in-memory big-data processing [8]. Ahn et al. [8] tuned a PIM accelerator for large-scale graph processing called Tesseract. Tesseract is equipped with efficient communication scheme to hide long access latency between different cores and to assure the atomic memory update operation. Regarding to the specialized memory access characteristics of graph processing workloads, new hardware prefetchers are utilized in Tesseract to enhance the utilization of memory bandwidth. Tesseract can improve the delivered performance by around one order of magnitude compared to the baseline system while reducing the energy consumption by 87% on average.

The Rollback-Free Value Prediction (RFVP) technique is one of the approximation techniques that alleviates the memory wall through approximation of the load value [176]. However, due to the nature of prediction methodologies in which the predicted value can be inaccurate, thus a sensitive applications data structure can be contaminated.

Farmahini-Farahani et al. [51] proposed to apply 3D-stack technology for connecting Coarse-Grain Reconfigurable Accelerators (CGRAs) on top of off-chip DRAM units using Through-Silicon Vias (TSVs). This schematic, referred as DRAMA, aims to offload data-intensive operations to CGRAs while the processor executes the remaining parts of the program. The main reason is that the processing of data will be taken place much closer to memory units which results in significant reduction of the cost required for data movement. The Near-DRAM Acceleration (NDA) architecture proposed by Farmahini-Farahani et al. [52] also enables processing data using CGRAs which are 3D-stacked atop DRAM devices. NDA architecture requires trivial change in DRAM device's I/O circuit while host processor design remains unchanged. NDA can achieve 1.67X higher performance than the existing accelerators within the processor while consuming 68% lower energy for data transfer.

Guo et al. [67] proposed a novel PIM-based architecture, referred as AC-DIMM, to employ associative search engine along with processing in memory. To maximize the AC-DIMM throughput, associative TCAM accelerator implemented by STT-MRAM devices are used while the store of key-value pair is optimized to be located in the same TCAM row. These considerations enable AC-DIMM to outperform conventional RAM-based system by around 4.2X while consuming 6.5X lower energy.

The comparative advantages of the different approaches to tackle the memory performance bottleneck by leveraging techniques such as near-memory processing [132, 133], accelerator-based hardware architectures [59], and approximate computing [45] to reduce data transfer cost are listed in Table 2.2.

Table 2.2: Comparison of state-of-the-art techniques for reducing off-chip data transfer.

Methodology	**Data Quality**	**3D-stacked Integration**	**Memory Subsystem Overhead**	**Beneficial for Irregular Access Pattern**
RFVP [176]	approximate	N/A	prediction stage	no
PEI [9]	exact	yes, with DRAM	dedicated processor	restricted to temporal locality
NDA [49]	exact	yes, near-DRAM	coarse-grained	yes
Tesseract [8]	exact	yes, with DRAM	fixed bank design	yes, for graph-structured data

CHAPTER 3: BIT-UPSET VULNERABILITY FACTOR ANALYSIS FOR EDRAM-BASED LLC

In order to preserve data integrity in eDRAM, the charge in each capacitor must be refreshed periodically. This means the data stored in eDRAM is required to be read out and written back into cell by each refresh operation. This new lifetime sequence needs to be investigated to accurately reveal how different lifetime sequences of cache data contribute to vulnerability. Thus, despite the fact that the most previous works studied the trade-offs between performance, energy consumption and achieved cache reliability, more systematic study of cache vulnerability is still needed to present a detailed lifetime model for data arrays in eDRAM LLC.

Given these challenges, we propose a more accurate and application-relevant lifetime model while taking the refresh operation scheme for LLC into consideration. The proposed lifetime model distinguishes among six lifetime sequences for each data item located in eDRAM LLC according to the previous activity and current one. These sequences are further categorized into two groups, $vulnerable$ and $nonvulnerable$ sequences. A vulnerable sequence is characterized by the fact that any error that is encountered during this sequence has the potential to be either consumed by the CPU or committed to memory by a write. However, if the exposure occurs during nonvulnerable sequence, no program failure will result.

The proposed model relies on LLC ensemble behavior analysis using trace files obtained from PARSEC benchmark suites running on an extended version of MARSSx86 [29]. The results of our analysis can be used to develop efficient protection techniques for high reliable eDRAM cache design.

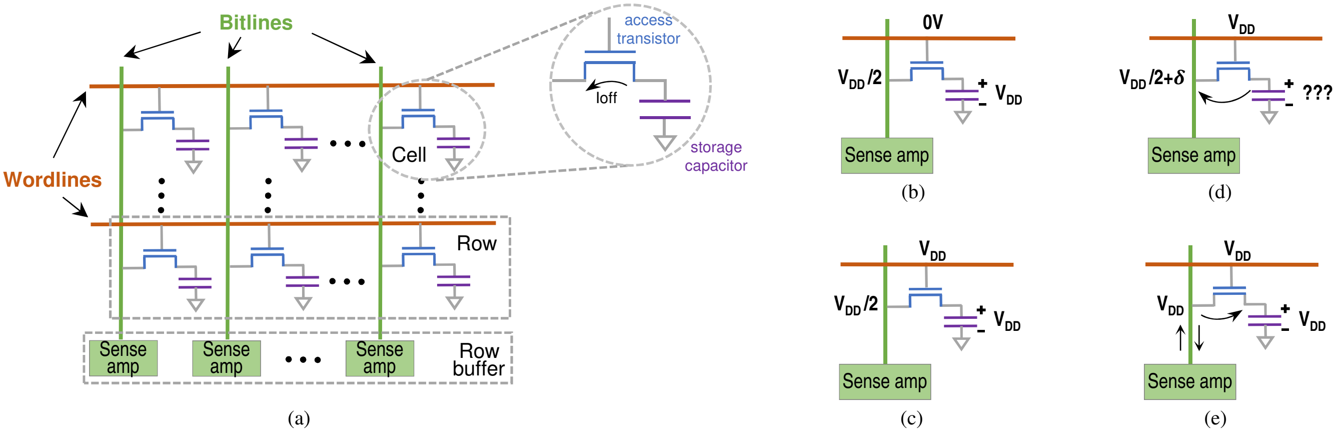


Figure 3.1: Typical eDRAM system organization, (a) eDRAM bank structure, (b) Precharged state, (c) Wordline raised, (d) Charge sharing, (e) Sensing.

3.1 eDRAM Cell Organization and Operation

The eDRAM cells are organized in a two-dimensional arrays called $bank$ as illustrated in Fig. 3.1 (a). Each eDRAM cell consists of a dedicated transistor which connects the $bitline$ wire to its associated capacitor. The access transistor in a row are connected and controlled by a wire called $wordline$.

The following steps are required to access the data stored in each row:

- First, all bitline wires in the bank need to be $precharged$ to $V_{DD}/2$ as illustrated in Fig. 3.1 (b).
- As shown in Fig. 3.1 (c), the row is activated with wordline overdriven to V_{DD} which follows by connecting all capacitors in the row to their corresponding bitlines.
- In the $Charge\ Sharing$ process, the charge either flows from capacitor to bitline or vice versa according to the amount of charge stored in capacitor.
- The voltage change in the bitline is detected and enlarged using the respective connected

sense amplifier. As the final step shown in Fig. 3.1 (e), the sense amplifier drives the bitline entirely either to V_{DD} or 0 V.

Therefore, the cells of the enabled row can be read by sensing or be written by driving the voltage on the associated bitlines.

3.1.1 eDRAM Retention Time for Different Technology Node

The charge of capacitor is lost overtime through access transistor which makes the refresh operation inevitable for eDRAM celsl. The threshold voltage (V_{th}) of access transistor plays an important role to determine the retention time of eDRAM cell. The high V_{th} leads to lower leakage which means the eDRAM cell retains state for longer time while low V_{th} provides less retention time due to higher leakage power. The retention time of an eDRAM cell is defined as the leakage time at which the capacitor loses α portion of the stored charge [98] and can be defined as below:

$$T_{ret} = \frac{\alpha}{\beta} \tag{3.1}$$

where α is the allowable amount of storage charge which capacitor can lose and β is the off drain current through access transistor. Kong et al. [98] showed that the eDRAM cell can stay operational while α equals to 6/10th of the stored charge. The T_{ret} can also be expressed in Eq. 3.2 which comes from [98] and [4].

$$T_{ret} = \alpha \times \frac{L}{W} \times 10^{V_{th}/S_{th}} \times 10^{9}/300sec \tag{3.2}$$

where W and L are the width and length of the access transistor channel, and S_{th} is subthreshold slope. The detail of Eq. 3.2 can be found in [4].

As the technology scales down, the V_{th} of access transistor reduces accordingly. Hence, if the V_{th} is substituted with a smaller V_{th} in Eq. 3.2, the eDRAM retention time is reduced. This phenomenon has been shown in Fig. 3.2 in which the probability of a retention failure in eDRAM cell fabricated by $32nm$ technology node is higher than 45nm eDRAM cell (derived from Figure 3 in [98]).

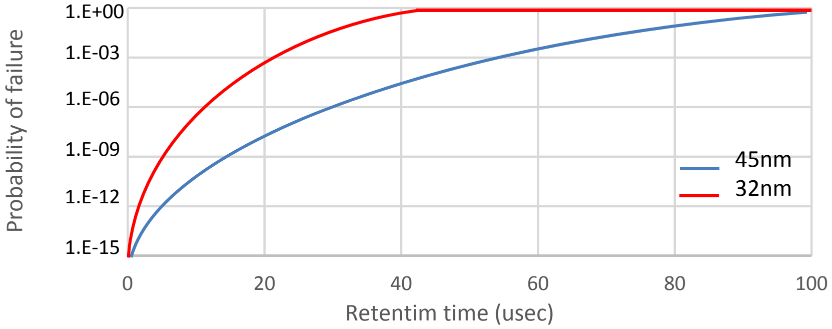


Figure 3.2: eDRAM retention time distribution for 45nm and 32nm technology nodes.

3.1.2 Single Event Effect

The Single Event Effect (SEE) is induced by the interaction of the high-energy particles with electronic components as illustrated in Fig. 3.3. The generated electron and hole pairs are collected into cell junctions through the so-called funneling mechanism. Most of the charge is obsorbed at the junction via a deformation of the junction potential. The remaining charge is diffused and collected in the substrate. The access transistors in eDRAM module are sensitive to the particle′s radiation because the stored data is required to be blocked when no read or write is operated. A particle can induce a SEU when it strikes at the drain region of an off-MOSFET. The voltage at the struck node drifts and turn the radiation-induced current into a voltage transient because the released charge is collected at the reverse-biased drain p-n junction. If the drain potential is smaller than the cell switching voltage, the current decreases the potential at the drain node and likely changes the initial state [16][175]. Radiation-induced Soft Error Rate (SER) can be expressed as:

$$SER \approx Area \times exp^{Q_C/Q_{Coll}} \tag{3.3}$$

where $Area$ is diffusion area of collected charges which is linearly proportional to the cell size of eDRAM, Q_C is the minimum collected charges incurring soft error and Q_{Coll} is the collected charges compiled by drift (Q_{drift}) in depletion layer and diffusion (Q_{diff}) from the silicon substrate. As the size of cell junctions shrinks in eDRAM, the amount of charge collected decreases due to lowered-depth of sensitive depletion region. However, the amount of Q_C decreases even more rapidly due to the power supply reduction that is used for increasing the performance of succeeding generation of eDRAM. Thus, the SER exponentially increases for a new generation device activated by lowered power supply. Empirical data is presented in [47].

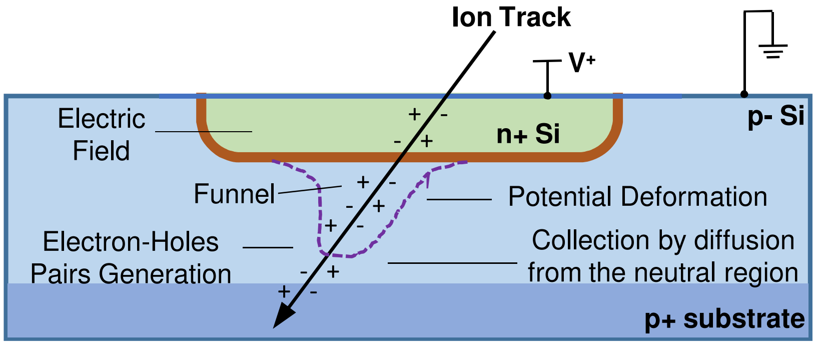

Ion Track
V+
Electric Field
n+ Si
p- Si
Funnel
Potential Deformation
Electron-Holes Pairs Generation
Collection by diffusion from the neutral region
p+ substrate

Figure 3.3: The impact of ionizing particle in a reverse-biased p-n junction.

3.1.3 LLC Reference Characteristics

The eDRAM LLC is shared by all on-die cores to offer worthwhile benefits for such workloads exhibiting significant data-sharing. As illustrated in Fig. 3.4 and 3.5 , the time varying behavior of each benchmark shows multiple sequences of cache reliability issue where some sequences are vulnerable and some sequences that are not. For example, the $streamcluster$ benchmark is a read intensive workload in which a high proportion of shared cachelines are accessed by consecutive read memory operations. Such a cache line generation information can be exploited to show the correlation among the LLC access pattern and the cacheline vulnerability factor.

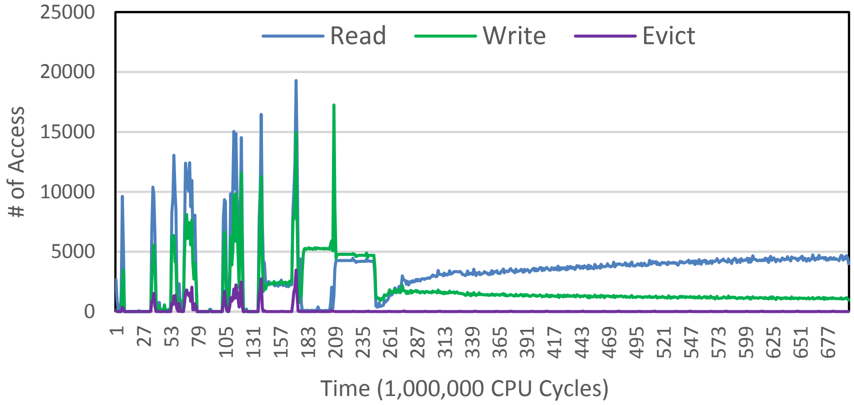


Figure 3.4: LLC memory access for canneal.

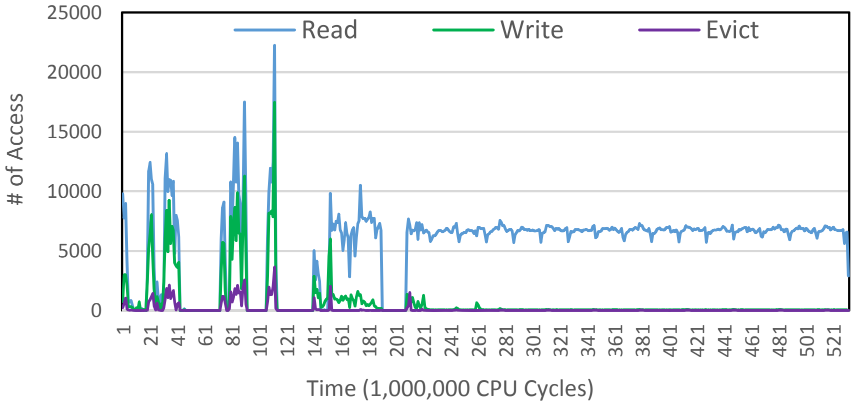


Figure 3.5: LLC memory access for streamcluster.

3.2 The Proposed eDRAM Vulnerability Estimation Algorithm

3.2.1 A general Lifetime Model for Data Array

A cache line is brought into LLC on a read or write miss. Then, it will be accessed either by reads or writes for a couple of times, and finally, it will be replaced by a new cache line. According to the existing LLC reliability analysis methodologies [168][114], the lifetime of a cacheline can be categorized into the following sequences:

- Read-Read (RR): the lifetime between two consecutive reads
- Write-Read (WR): the lifetime of a write operation up until its first read
- Write-Evict (WE): the lifetime between the last write and the cache line replacement by a new cacheline
- Read-Evict (RE): the lifetime between the last read before cache line replacement
- Read-Write (RW): the lifetime between the last read before the write operation
- Write-Write (WW): the lifetime between two consecutive writes

A *lifetime sequence* is defined as *vulnerable sequence* if an error may propagate out of cache, either to the CPU or to the lower level of memory hierarchy. Apparently, the first four intervals, RR, WR, WE, and RE can be considered as vulnerable sequences. The reason for this is that if any error occurs in aforementioned sequences, it has this potential to be either read by CPU or committed to the memory. On the other hand, the RW and WW sequences are *nonvulnerable sequence* in which if an error occurs, it is simply masked off through overwritten operation, presenting no program failure.

3.2.2 The Proposed Lifetime Model for eDRAM Data Array

The critical charge of eDRAM is dependent on write voltage, storage and bit-line capacitance and the minimum voltage difference required by the sense amplier [153]. These factors make eDRAM cell vulnerable against soft error for a particular portion of lifetime sequence unlike SRAM cell which is vulnerable for the entire sequence period. There are three kinds of SER modes for eDRAM cell wherein, if any exposure occurs it may result in potential errors:

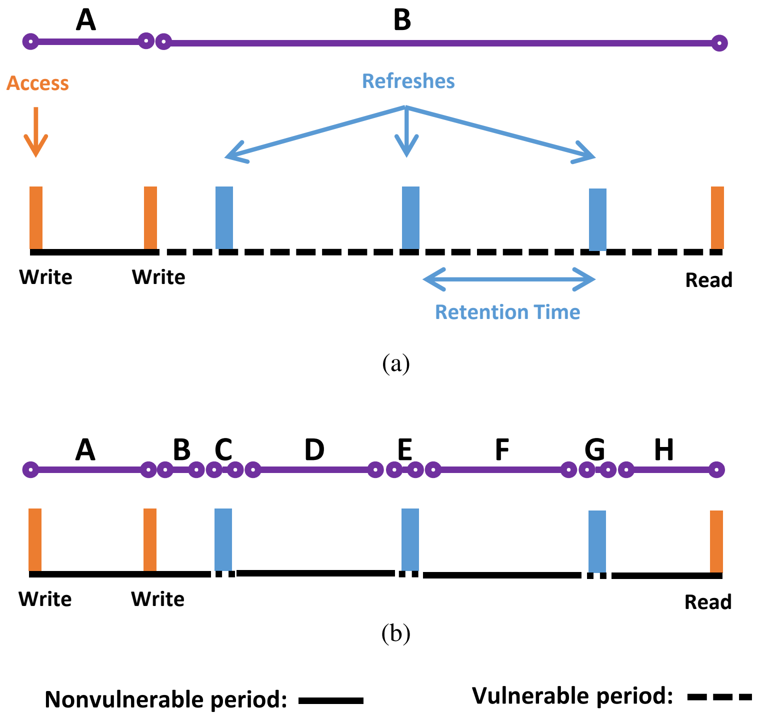
- Memory mode: This SER mode is the consequence of the injection of a high-energy particle into an eDRAM cell when logic 1 is stored in the storage capacitance.
- Bit mode: The soft errors are produced in this mode if SEU strikes a bit-line junction during the bit/bit-bar floating time and logic 1 is stored in the cell to be read.
- Bit-bar mode: If a SEU is injected into bit-bar junction during bit/bit-bar floating time and logic 0 is stored in the cell to be read, it can produce bit-bar mode soft errors.

The error produced is vulnerable exclusively if it occurs in abovementioned general lifetime sequences because it is either consumed by CPU or committed to the memory. Thus, to estimate the vulnerable sequence of eDRAM, it is necessary to integrate the existing lifetime model with the SER modes for eDRAM. In order to estimate the BUVF, all three kinds of SER modes for eDRAM cell must be considered. The memory and bit modes soft errors are taken place when the storage node has a logic 1 value while bit-bar mode occurs when the eDRAM cells contains logic 0 value.

To estimate BUVF for eDRAM cell containing logic 1 value, the entire lifetime sequence can be considered as vulnerable sequence if the second memory operation in two consecutive accesses to that data item is either read or evict. As shown in Fig. 3.6 (a), the *sequence A* is a WW sequence which is not vulnerable against soft errors. The *sequence B* represents the vulnerable

WR sequence in which the last write operation is following by three refresh operations and then by a read operation.

On the other hand, the BUVF for eDRAM cell containing logic 0 value is estimated according to the bit/bit-bar floating time allocated in vulnerable sequence. As shown in Fig. 3.6 (b), *sequence C*, *E* and *G* are vulnerable sequences where the eDRAM cell is in its refresh mode when logic 0 is stored in it. Even though this vulnerable sequence may seem negligible, it shares a significant portion of BUVF when we take the large size of eDRAM and number of refresh operations originating within each vulnerable sequence into consideration.



(a)

(b)

Figure 3.6: The lifetime of eDRAM cell. (a) the logic 1 is stored in the cell, (b) the logic 0 is stored in the cell.

3.2.3 Bit-Upset Vulnerability Factor Analysis of eDRAM Data Array

The cache BUVF introduced in this work is defined as the average rate of data items in vulnerable sequences over the total data items that cache can accommodate during the execution. BUVF can be calculated as follows:

$$BUVF = \frac{(vdi(0) + vdi(1))}{\sum(data\_items \times exec\_time)} \tag{3.4}$$

where $vdi(0)$ and $vdi(1)$ are the vulnerable period for eDRAM cell containing logic value 0 and 1, respectively. Thus, BUVF can be calculated as the summation of $BUVF_0$ and $BUVF_1$ for eDRAM cell storing logic value 0 and 1, respectively. The $BUVF_0$ can be calculated as:

$$BUVF_0 = \frac{\sum\limits_{i=1}^{n}(data\_item(0)_i \times ref\_dur \times \sum\limits_{j=1}^{l} vul\_sequence_j)}{\sum(data\_items \times exec\_time)} \tag{3.5}$$

where $data\_item(0)_i$ is the i^{th} eDRAM cell storing logic 0 value, ref_dur is the period time for each refresh operation and l represents the number of refreshes occurring in the duration time of j^{th} vulnerable sequence for $data\_item(0)_i$. Meanwhile, the $BUVF_1$ is:

$$BUVF_1 = \frac{\sum\limits_{i=1}^{n}(data\_item(1)_i \times \sum\limits_{j=1}^{k} vul\_sequence_j)}{\sum(data\_items \times exec\_time)} \tag{3.6}$$

where $data\_item(1)_i$ is the i^{th} eDRAM cell containing logic 1 value, $vul\_sequence_j$ is the duration time of j^{th} vulnerable sequence for $data\_item(1)_i$ and k represents the boundary which is the end of memory access to $data\_item(1)_i$.

A high value of BUVF for a data item located in eDRAM LLC indicates reduced resiliency to soft errors. Therefore, we recast the problem of reliable eDRAM LLC design as a straightforward search for reduced BUVF.

3.3 Experimental Setup

We analyze LLC ensemble behavior using traces from an extended version of MARSSx86 [29]. The traces capture access type to LLC which are typically read, write, and evict. We model a Chip MultiProcessor (CMP) with eight single-threaded x86 cores. Each core has private L1 and L2 caches and the LLC is shared among all the cores to increase the performance. The detail of our model can be found in Table 5.3. Twelve applications from the PARSEC benchmarks suite are selected and executed 500 million instructions starting at the Region Of Interest (ROI) after warming up the cache for 5 million instructions. Furthermore, the simsmall input sets are used for all PARSEC applications.

We evaluate the proposed scheme for all vulnerable sequences. However, RR and WR are the dominant contributors to the BUVF of LLC data array because the large size of LLC provides long-term residency for data items in LLC which results in the remained vulnerable sequences to be less active during program execution. Accordingly, the results of data vulnerability analysis in two major contributors to the BUVF are described in this paper.

Fig. 3.8 (a) shows the proportion of data pieces resided in vulnerable sequences. We categorized each vulnerable sequence into three lengths based on the time interval between two consecutive accesses to data: *1) Short, 2) Medium, 3) Long*, denoted as SRR/SWR, MRR/MWR, or LRR/LWR, respectively. The short duration sequence indicates data instances with an interval less than 1 million cycles while this interval increases from 1 million to 50 million cycles for medium sub-vulnerable sequence. The data elements with intervals exceeding 50 million cycles are considered as long duration sequence. For example, the rightmost orange column for each benchmark represents the long-term WR sequence which averages 0.23% across the suite. The second green column from the left shows the RR elements with medium duration interval contributing the largest share to the BUVF.

Fig. 3.8 (b) illustrates the sequence distribution of data instances in eDRAM LLC. The vulnerable sequences account for about 27.24% of data array lifetime in the cache, among which RR contributes about 23.45%. The data instances in MRR and LRR sequences are dominate the overall RR time, 23.14% on the average across the entire suite. Although the second largest set of vulnerable data are elements located in SRR sequence, this portion of cache space contributes the second least vulnerability to BUVF which is the result of spending a small fraction of program execution time in SRR sequence. This observation confirms that the long-term residency of a data block in LLC between two consecutive accesses significantly increases its vulnerability to soft errors. The profile results convince us that the read intensive benchmarks with medium and long read-read instances, i.e. $streamcluster$, contribute the most BUVF in the data cache.

The results obtained by BUVF characterization and analysis are shown in Fig. 3.9. A smaller value of BUVF implies that the cache is more resilient against soft errors. The BUVF values for $raytrace$ and $facesim$ benchmarks are 0.020 and 0.022, respectively, indicating the high resiliency of them to soft errors. The main reason behind the reduced BUVF value for these benchmarks is that the data remained for a shorter period within the vulnerable sequences. On the other hand, the BUVF value of $streamcluster$ is 0.76 which is the highest BUVF among benchmarks under study.

Even though the data located in LRR sequence only account for 0.73 percent of total data items as shown in Fig. 3.8 (a), this sequence is the second largest contributor to BUVF. Furthermore, the two potential duration sequences MRR and LRR together account for 0.16 of vulnerability factor across all workloads, pointing out that the RR sequence contributes the most to BUVF. This agrees with typical distribution of data access being read-predominant versus write-perdominant.

Table 3.1: Simulator configuration

Processor	3GHz processor Fetch/Exec/Commit width 4
Private L1-I/D	SRAM, 32 KB, 8-way set assoc., MESI cache
Private L2 Conf.	SRAM, 512 KB, 8-way set assoc., MESI cache
Shared L3	eDRAM, 96 MB, 16-way set assoc., 16 bank, WB cache
Main memory	8 GB, 1 channel, 4 ranks/channel, 8 bank/rank

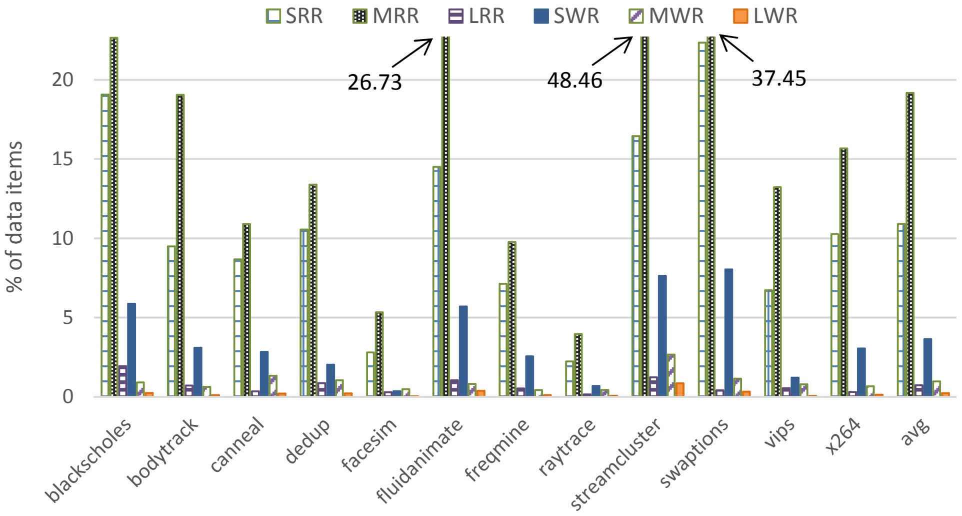


Figure 3.7: Distribution of the proportion of data items in vulnerable sequence

3.4 Conclusions

In this paper, we conducted a detailed study on eDRAM LLC vulnerability to soft errors along with a new lifetime modeling method. The proposed BUVF model investigates three SER modes for eDRAM cells in which the minimum collected charge has the potential to induce soft error. The Memory, bit and bit-bar SER modes are identified and integrated into the proposed BUVF to accurately capture the various kinds of possible soft errors in eDRAM cell. We evaluated our model using LLC ensemble behavior traces obtained from MARSSx86 running PARSEC 2.1 applications. The major contributors to the LLC vulnerability are identified according to the results obtained from our BUVF analysis. Our results indicate that the two potential duration sequences MRR and LRR together contribute the most to BUVF, and thus are the primary sources of the instability within LLC on a multi-core processor die.

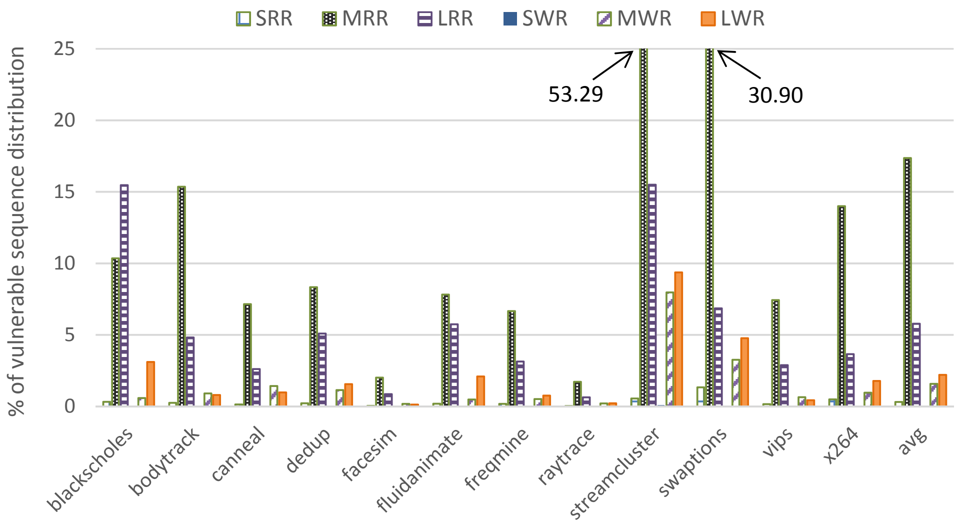


Figure 3.8: The sequence distribution of data items

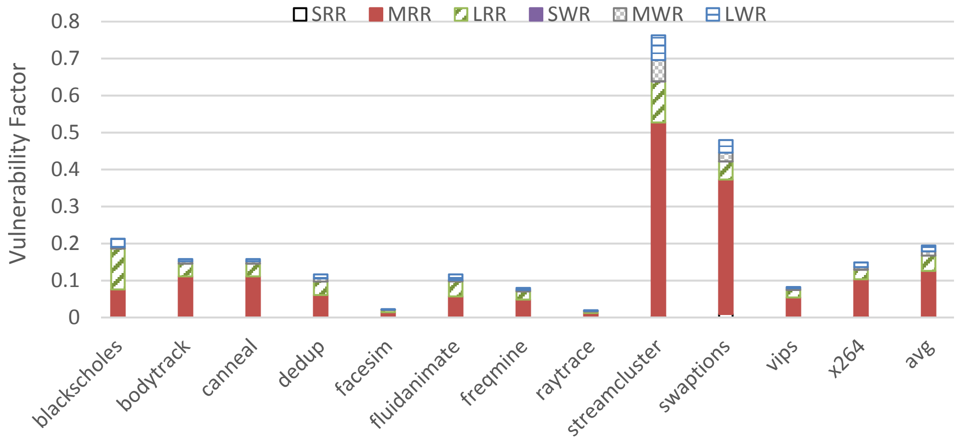


Figure 3.9: Measured BUVF behavior for the workloads of the PARSEC suite.

CHAPTER 4: VARIATION-IMMUNE NVM-BASED SYSTEM DESIGN

The continuous demand for increasing the data throughput of the overarching system entails the cooperation of billions of CMOS transistors in a small-scale die area. The scaling trends empowers rising device density by introducing highly accurate fabrication devices which are capable of manufacturing low power systems with deeply-scaled transistors. Even though significant effort has been invested in ensuring the precise fabrication of nanoscale CMOS transistors, the technology scaling has confronted the limitation of the scalability in conventional semiconductor technology due to increasing the static power consumption and exacerbation of the reliability and Process Variations (PV) issues.

The work herein is motivated by the observation that around 27.5% of the sensed data in the PARSEC suite when utilizing a STT-MRAM based Last Level Cache (LLC) has the potential to be read incorrectly due to PV. Even though up to 6% on average of the incorrectly sensed data will be overwritten prior to be consumed by processor or to be committed to the main memory, which reduces the urge of their accommodation, there is still a significant portion of incorrectly sensed data that must be handled before manifesting themselves as wrong output, or application crash, or prolonged program execution [91]. This observation inspired a redesign of the schematic of SAs array in STT-MRAM data arrays to provide improved sensing margin.

To provide reliable sensing operation while taking the energy budget into consideration, we propose a circuit-architecture cross-layer solution suitable for multi-core processors as well as IoT devices. Our proposed technique, referred to as *Self-Organized Sub-bank (SOS)*, partitions STT-MRAM data arrays into several sub-banks to directly access requested data while introducing individualized sensing resolution. Sub-banks are evaluated and tagged during an initial *Power-On Self-Test (POST)* phase to identify the preferred SA for that particular sub-bank. To be specific, if the error rate of the impacted NVM cells in a sub-bank exceeds the predefined threshold, a High Resilience SA is assigned which is an adapted SA from the proposed SA in [85]. Otherwise, a Low Power Delay Product SA offering reduced delay and power consumption is assigned which is adapted herein from the proposed SA in [186].

4.1 The Spin-Transfer Torque Magnetic Random Access Memory (STT-MRAM)

As illustrated in Fig. 4.1, STT-MRAM utilizes MTJ device as storage element in which a thin insulating oxide later, e.g. MgO, is sandwiched by two ferromagnetic layers [85]. Moreover, the upper ferromagnetic layer is usually aliased as the free layer having a polarity of magnetic field which can be flipped during a write operation. Meanwhile, the lower ferromagnetic layer usually referred to as the pinned layer is designed to have its magnetization fixed. Thus, MTJs have a low (high) resistance distribution if the magnetization of the free layer and the fixed layer are parallel *P* (anti-parallel *AP*). Accordingly, low (high) resistance distribution is stored in MTJ, instead of traditional electronic charge or current flow.

For a read operation, a small current is required to be driven from bit-line to source line. On the other hand, a successful write operation requires a current flow drive either from bit-line to source-line or vice versa, depending on the differential voltage between these two lines. Although STT-MRAM does not suffer from significant write endurance, however the advent of long write latency and high energy consumption exacerbate the reliability implications of STT-MRAM.

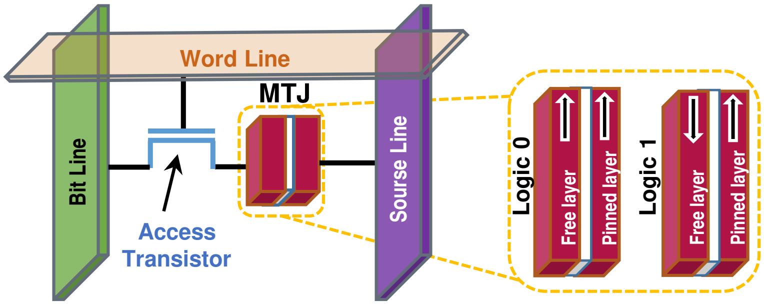


Figure 4.1: An illustration of a 1T1R STT-MRAM cell embedded in the cache organization.

4.1.1 Process Variation in STT-MRAM

The imprecise fabrication process in nano-scale technology results in Process Variations (PV). The impact of PV on CMOS-based logic circuit and memory storage is a combination of *systematic variations* which are mostly caused by lithographic aberrations [55] [185] and *random variations* induced by random doping fluctuations [185] [21]. The effect of systematic variations is spatially correlated among the effective gate length, L_{eff}, while the threshold voltage, V_{th}, is influenced by random variations. The MTJ resistance is primarily changed due to PV impact on MgO thickness and MTJ shape [159]. In our PV model, we assume that the cache tag and peripherals (e.g., row decoder, column decoder, row buffer and SAs) are fabricated at the CMOS layer while memory cells are realized through MTJ devices. Since the MTJs are vertically stacked on top of the CMOS layer and these components are tightly coupled to realize the function of STT-MRAM, but the read sensing margin varies readily based on the effect of PV on that particular region of the die. Accordingly, we consider the same PV parameters to model both CMOS and MTJ variations in VARIUS [150] which is based on static analysis tool R [1] and geoR package [143].

4.1.2 Extracting the PV Parameters

As listed in Table 4.1, we choose our σ/μ for L_{eff} variation to be 10% to align with the previous measurements reported in [107]. We consider $\phi = 0.5$ based on [55] which means the gate length has a spatial correlation range close to half of chip's width [163]. We randomly select one map among a large pool of maps which are generated by VARIUS with resolution of one million (1000×1000) sample points. The degree of variation is shown by a range of colors. Each color corresponds to a specific value of sample points as shown in Fig. 4.3. In our simulation, we consider the amount of PV for each site based on the location of the LLC components within the floorplan and their associated sample points. Thus, VARIUS generates a relatively accurate estimation of the impact of PV on the read sense margin of each sub-bank data array.

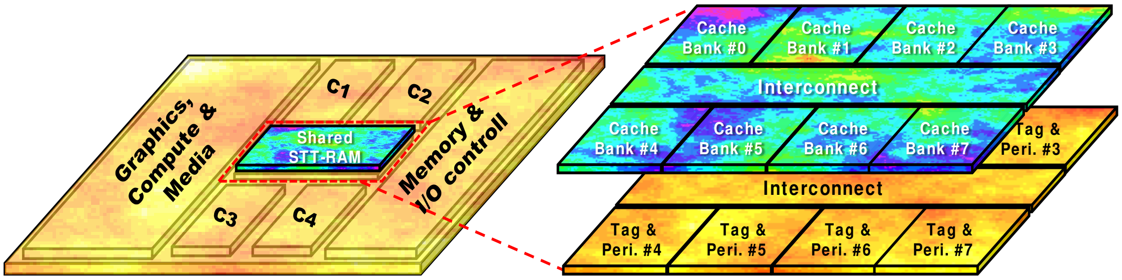


Figure 4.2: PV map of a 4-core CMP die.

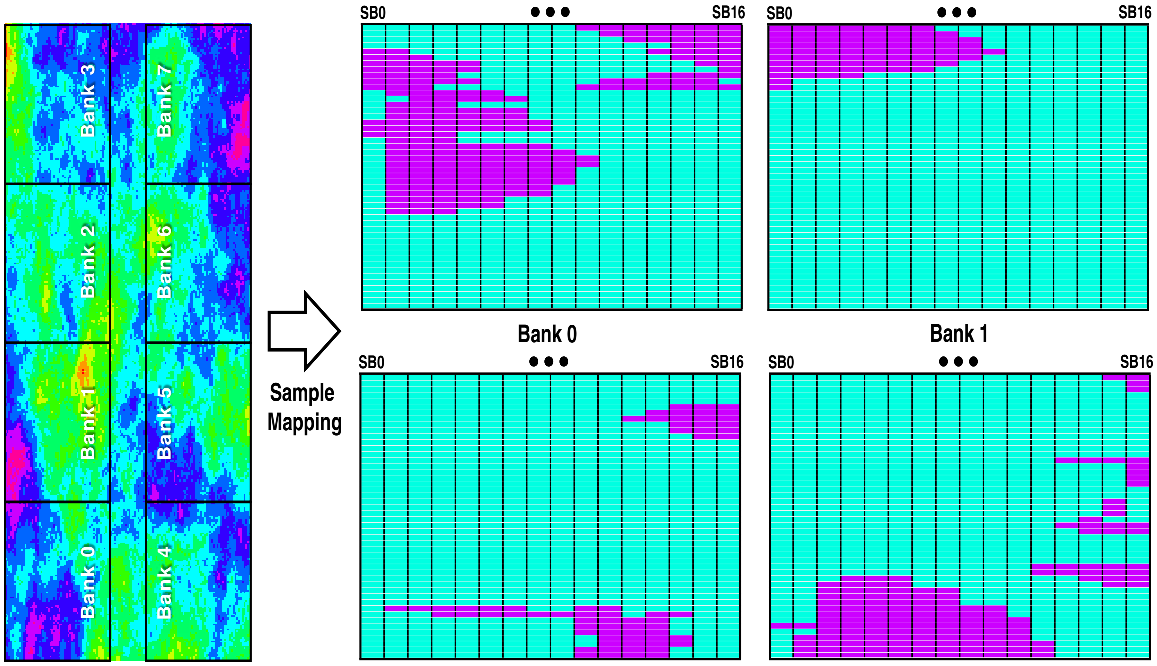


Figure 4.3: Determining the preferred SA circuit based on post-fabrication sub-bank resiliency assessment to PV.

4.1.3 Power-On Self-Test (POST)

As shown in Fig. 4.2, the cache bank floorplan of STT-MRAM layer is superimposed on the map. In our SOS approach, each cache bank is partitioned into 16 sub-banks. The size of each sub-bank is matched with the word size to maintain the energy consumption of the tag to be as low as possible, e. g. 32-bits in our case study. We consider one additional bit per each sub-bank to identify the preferred SA for that particular sub-bank during post-fabrication resiliency assessment to PV. The POST phase is basically a March Test that targets PV-induced faults in STT-MRAM [31]. We assume the proposed SRAM March Test with *O(n)* test length can be utilized for our purpose because the tag and peripherals of STT-MRAM are considered to be implemented in CMOS layer. Thus, variation-induced delay fault in both SRAM and STT-MRAM manifests itself as the same fault model such as insufficient pre-charge time, insufficient sense time, insufficient amplify time, disturbance of sense operation, and simultaneously activation of multiple word lines.

In this regard, PV-aware March Test traverses all STT-MRAM data arrays and performs a sequence of operations (e. g., exhaust all pair-wise address transitions) to identify PV-induced delay faults in each cell [31]. If the error rate of the impacted STT-MRAM cells in a sub-bank exceeds the predefined threshold, the extra bit is set to '0' indicating that an array of reliable SAs are required for sensing data array of this sub-bank. Otherwise, the extra bit is set to '1' which indicates that an array of low-power SAs offering reduced delay and power consumption can be considered for that particular sub-bank. Since POST is a one-time operation, then it will not impact the performance of the memory as a whole resulting in incurring a negligible overhead. Taking advantage of this feature, we will be able to analyze memory cells before initiating the main stream operation, which identified those cells suffering the most from PV.

4.2 Self-Organized Sub-banks (SOS)

4.2.1 SOS Schematic for SAs Array Assignment to each Sub-bank

Among the contemporary proposed low-power and reliable SAs, we select two promising SAs that exhibit remarkable improvement in either reliability or energy consumption: 1) PCSA: the low-power SA circuit that incurs very low energy consumption compared to other designs [186], and 2) SPCSA: the high reliable SA circuit that offers approximately 2.5-fold reduced error rate compared to PCSA [85]. Herein, we call the ensemble of PCSA, SPCSA and MUX as a Merged Sense Amplifier (MSA) which utilizes each device's properties to increase the performance and reliability of the memory.

The schematic of SOS design is depicted in Fig. 4.4 and the process of assigning preferred SA to each sub-bank is shown in Algorithm 1. After POST process and determining the preferred SA for that particular sub-bank, a select input is used in the circuit called **MODE** to choose between the two SAs based on assigned bit set value as shown in Fig. 4.4. Since PCSA consists of fewer CMOS transistors, it offers enhanced performance in terms of sensing delay and Energy Delay Product (EDP) compared to SPCSA. In the branch containing the main MTJ in PCSA, we have four transistors namely MP0, MP1, MN0, and MN4 and in the branch that includes reference MTJ we have also four transistors MP2, MP3, MN1, and MN4 as depicted in Fig. 4.5. However, the main MTJ branch in SPCSA consists of two transistors MP0 and MN4 and two transistors MP5 and MN4 in the reference MTJ branch which makes it less vulnerable to PV, as shown in Fig. 4.5. This redesign of the SA introduces an elevated sense margin which in turn results in decreased Bit Error Rate (BER). Nonetheless, the reduced BER comes with the cost of utilizing greater number of transistors in SPCSA design which incurs higher EDP.

The **MODE** signal controls the operation mode of the circuit to either operate in PCSA mode or SPCSA mode. If the input **MODE** is asserted then the circuit will operate in PCSA mode. On the other hand, if **MODE** is de-asserted will change the operation of the circuit to SPCSA mode. In order to further reduce the PV effect on the reference cell, we use the configuration shown with red dotted region in Fig. 4.5 for MTJ0 which consists of (MTJ$_P$+MTJ$_{AP}$)$||$(MTJ$_P$+MTJ$_{AP}$) that will result in an ideal reference cell in terms of resistance which is (MTJ$_P$+MTJ$_{AP}$)/2 and in terms of process variation immunity [86]. The transition waveforms of the output of all the designs are provided in Fig. 4.10 in which the two operations of the proposed SOS design are shown. In addition, we propose an alternative for MSA that further improves energy consumption and reliability due to PV. The Adaptive Sense Amplifier (ASA), as shown in Fig. 4.7, has a functionality similar to MSA. However, by utilizing the Energy Aware Sense Amplifier (EASA), as shown in Fig. 4.7 (a), and the Variation Immune Sense Amplifier (VISA), as shown in Fig. 4.7 (b), instead of PCSA and SPCSA it can achieve better energy and reliability profile respectively.

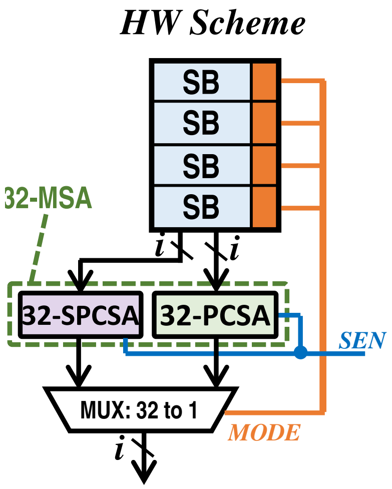


Figure 4.4: Proposed SOS.

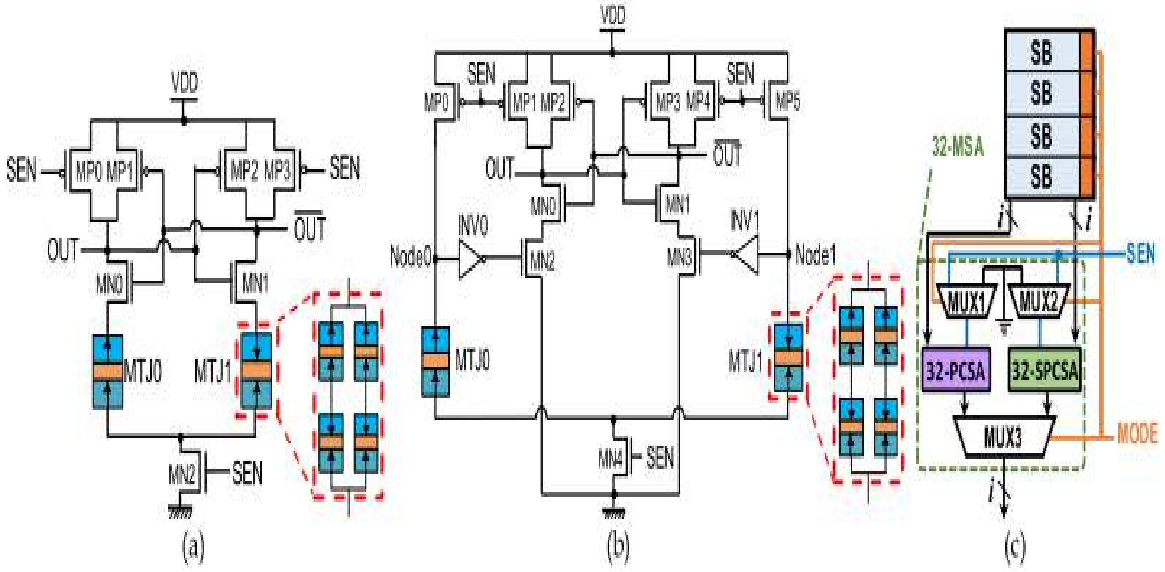


Figure 4.5: (a) PCSA circuit (based on [186]), (b) SPCSA circuit (based on [85]), (c) MSA.

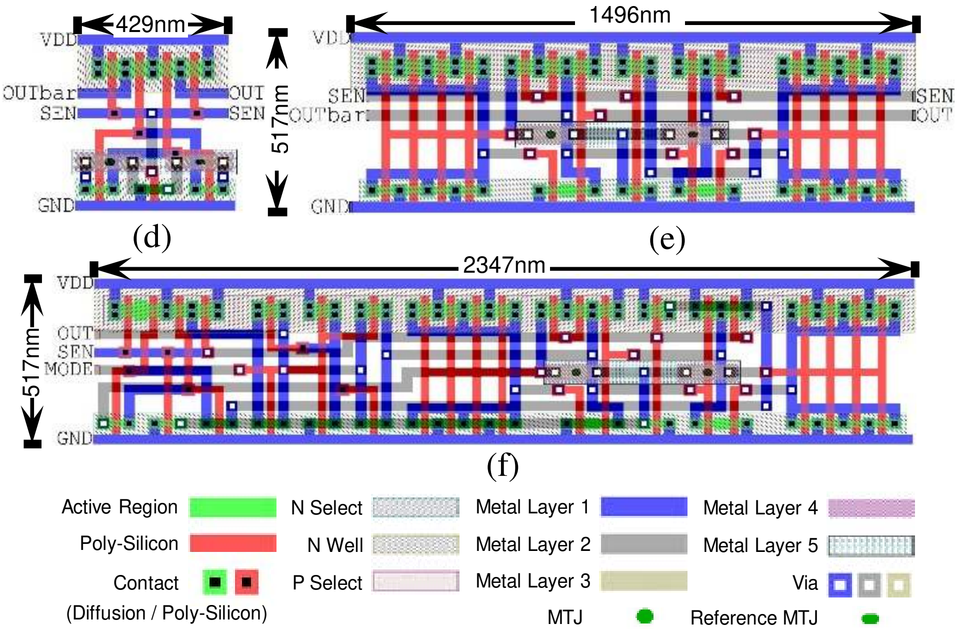


Figure 4.6: PCSA, SPCSA, and MSA Layout.

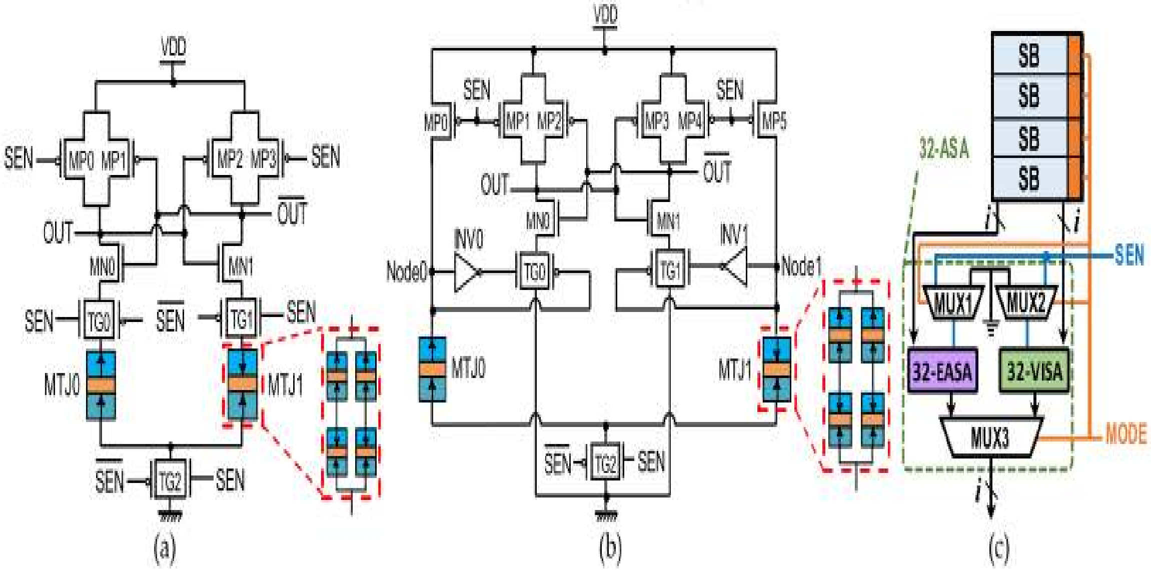


Figure 4.7: (a) EASA, (b) VISA, and (c) ASA.

Algorithm 1: SOS Approach to Assign Preferred SA to Sub-bank

```
Function SOS() /*SOS Approach for SA Assignment*/
for ∀ $cache\ line \in LLC$ do
    for ∀ $sub - bank \in cache\ line$ do
        begin
            POST() /*Power-On Self-Test*/
            Analyzer() /*Evaluate the correctness of the outputs*/

Function POST() /*Power-On Self-Test*/
begin
    set SEN = 1 /*start the sensing state*/
    if output != expected-value then
        ++number-wrong-outputs /*increase number of wrong outputs*/
    set SEN = 0 /*keep the sense signal in pre-charge state*/

Function Analyzer() /*Evaluate the correctness of the outputs*/
begin
    if number-wrong-outputs > threshold then
        set MODE = 0 /*assign SPCSA to sub-bank*/
        /*MUX takes sensed data from SPCSA to output*/
    else
        set MODE = 1 /*assign PCSA to sub-bank*/
        /*MUX takes sensed data from PCSA to output*/
```

4.2.2 Distribution of Bit Errors in a Sub-banks

If we assume that the sensing operation of a bit follows the binomial distribution, the probability distribution of exact number of incorrectly sensed *k* bits in a sub-bank can be calculated by the probability mass function, pmf. Herein, we incorporate pmf into Sub-Bank Bit Error Rate (*SBER*) which can be calculated as following:

$$SBER(k) = \binom{SB}{k}(BER)^k(1 - BER)^{SB-k}, k = 0, ..., SB \tag{4.1}$$

where *SB* is the number of bits in a sub-bank, *k* is number of incorrectly sensed bits, and *BER* is the probability that a bit is sensed incorrectly. However, since SOS considers the total number of incorrectly sensed bits in a sub-bank to determine which type of SAs array must be assigned to that particular sub-bank, the cumulative distribution function, cdf, is considered for decision making as follows:

$$SBER(X \leq k) = \sum_{i=0}^{k} \binom{SB}{i} (BER)^i (1 - BER)^{SB-i}, i = 0, ..., SB \tag{4.2}$$

For example, if we assume that BER is 0.5 for each bit in sub-bank and the number of bits resided in each sub-bank is 32, we have $SBER(X \geq 16) = 0.56$ which means the probability of incorrectly sensing more than 16 bits in a sub-bank is 0.56. The calculated SBER for sub-banks which are equipped with different versions of SAs are listed in Table 4.3 and Table 4.4. This theoretical assessment can be leveraged to estimate the proportion of BER in a sub-bank prior to running workloads for practical examination.

4.2.3 Fault Models Associated with Sensed Data

Independent of alternative fault models that can impact the stored value in STT-MRAM cell [32], overlooking PV effects during SA design may result in the sensed data differs from actual stored logic value while the read operation is taking place. This work concentrates on the faults that are caused by incorrect sensed data. To be specific, we classify the outcomes of SA operation to the following categories for broad adaption:

- **True Data Sensing (TDS):** The sensed data value is identical to the value stored in the STT-MRAM cell.
- **False Data Sensing (FDS):** The sensed data value differs from the value stored in the STT-MRAM cell. FDS can be further classified to vulnerable FDS (VFDS) and non-vulnerable FDS (NVFDS).
  - **Vulnerable FDS (VFDS):** The sensed false data propagates out of cache, either consumed by the process or committed to other levels of memory [91].

- **Non-Vulnerable FDS (NVFDS):** The replica copy of the sensed false data in the upper levels of cache will be overwritten by a write operation prior to being consumed. During a block eviction, replica data becomes written back to the lower levels of cache because it is a dirty victim block. Thus, this benign fault does not threaten the semantic correctness.

4.2.4 The Proposed PV/Energy -aware Cache Migration Policy

Several hybrid cache designs have been proposed in the past to improve the write performance while offering much larger cache capacity with low leakage power [159, 161, 148, 166, 82]. These methodologies have inspired us to maximize the efficiency of SOS by proposing a dynamic PV/energy -aware cache block migration policy that utilizes a mixed SRAM and STT-MRAM banks in LLC. Even though reliable SA offers high read sensing margin which results in high ratio of error-free read operations, it is still likely that the sensed data value differs from the value stored in the memory cell. To overcome this issue, we propose to transfer vulnerable read-intensive blocks to the ways that belong to low-PV impacted ways located in other STT-MRAM banks. The *not access intensive* blocks can still remain in their STT-MRAM based ways, whether they are high-PV impacted or not. To amortize the energy consumption and long bank service time due to write operation in STT-MRAM data array, we propose to allocate write intensive cache blocks from ways in SRAM banks. SRAM device offers both low dynamic power and high performance features for write operation which significantly improves the cache utilization and bank accessibility.

4.2.4.1 Hybrid SRAM and STT-MRAM LLC Design

Fig. 4.9 illustrates the scheme of a hybrid 8-way set associative SRAM and STT-MRAM LLC design, whereby way-0 and way-1 are implemented within SRAM-based banks while way-2 through way-7 are built in STT-MRAM based banks. This configuration has been selected based on our experimental results, whereby the average number of write intensive blocks in each set was approximately 2 across all workloads. Since the peripherals required for read and write operations in NVM arrays occupy relatively larger portion of cache footprint than peripherals required by SRAM arrays, it is beneficial to build the tag array with SRAM cells. Thus, in our design we assume that the entire tag array is built with SRAM. Unlike conventional cache design approaches where tag store and data array are accessed simultaneously to reduce access latency while incurring significant power overhead, we propose to split the cache access into two stages similar to the work presented in [166] but with adjustments in favor of high SOS throughput. If LLC is accessed with a read operation, the tag array and all STT-MRAM banks are accessed in parallel. Thus, assuming that data is found in STT-MRAM banks, the unnecessary accesses to SRAM banks can be skipped. Upon an LLC miss on STT-MRAM banks but hit on a tag corresponding to a SRAM bank, the associated data array of SRAM bank in LLC is accessed. Even though this mechanism incurs additional latency if the data is stored in SRAM banks, we argue that this incident occurs rarely since our insertion/migration policy maintains the read-dominant cache blocks in STT-MRAM banks while write-intensive blocks are transferred to SRAM banks. If the cache set is accessed by a write operation, the tag arrays and SRAM banks are searched in the first stage. If the data is not found in SRAM banks but found in a STT-MRAM bank, the corresponding STT-MRAM banks is accessed in the next stage. Unlike the insertion strategy in [4] where SRAM banks are selected for inserting fetched data from memory upon an LLC miss, our insertion policy allocates a way from either SRAM or STT-MRAM banks according to the miss type. In particular, the SRAM and STT-MRAM banks are allocated upon an LLC write miss and read miss, respectively.

Based on our observation presented in [92], a portion of a workload might be re-executed several times indicating that the read-intensive cache blocks which were brought to LLC once, transferred to low-PV impacted region of a set, and finally evicted need to be re-allocated from low-PV impacted STT-MRAM banks while being re-referenced again. In order to keep track of read intensive blocks even after eviction from LLC, we utilize a read-intensive block profiler which is basically a queue of 16 entries that maintains the address of recent frequently-read blocks. Upon a read miss on LLC, the address of missed data is searched in the profiler. If it is found, a cache block from low-PV impacted STT-MRAM ways based on Least Recently Used (LRU) is replaced by fetched data from memory. The dirty victim block is written back into memory while the clean victim block is silently dropped.

4.2.4.2 PV/Energy -aware Cache Migration Policy

Besides considering hybrid SRAM and STT-MRAM design to accelerate service to write operations and improve bank accessibility, we also propose an efficient block insertion/migration policy to maximize the SOS throughput as shown in Fig. 4.8. The tag store associated with STT-MRAM banks are equipped with three fields, Read Counter (RC), Write Counter (WC), and PV status. The main idea behind using RC is to identify vulnerable read-intensive blocks in the set. If a frequently-read block is allocated from a high-PV impacted STT-MRAM array, the cache block must be relocated to a low-PV impacted region of the set to guarantee the reliable read sensing operation. We conducted an extensive exploration to evaluate the preferred value for the read threshold level, NR_{th}, within our design. We found that if NR_{th} is small, the ratio of blocks that must be transferred to low-PV impacted region, significantly increases while if NR_{th} is large, then SOS utilization significantly decreases because only a few read-intensive cache blocks are selected for migration. Thus, we set NR_{th} based on extensive study on block access pattern of under test workloads. In addition, the not access intensive cache blocks located in low-PV impacted data array in STT-MRAM is selected to replace by vulnerable read-intensive block, if the corresponding RC of one of the high-PV impacted blocks reaches NR_{th}.

On the other hand, WC is a saturating counter to keep track of write access pattern to a cache block. If WC reaches its NW_{th} , write threshold level, it is considered as a write-intensive block. We propose to transfer these blocks to SRAM data array to amortize the latency and high dynamic energy consumption associated with incoming write operations. The PV status determines that whether a cache block is located in low-PV impacted region or in high-PV impacted data array. This bit is set based on a consensus decision-making process in the tag store during POST phase where a PV-aware March Test traverses sub-banks of cache block to identify the PV-impacted sub-banks. Fig. 3 illustrates an example of migration policy for a read-intensive block located in high-PV impacted region of STT-MRAM cache. Upon a read hit on way-2 in STT-MRAM bank, the RC reaches its NW_{th}, indicating that it is highly possible that the incoming accesses to this block is read-dominant operation. To reduce the probability of incorrectly sensing the stored value in STT-MRAM, the proposed migration policy swap the selected read-intensive block resided in high-PV impacted region with a not access intensive block located in low-PV impacted region based on LRU stacks in the tag array. A swap buffer is employed to properly enable the block transfer between low-PV impacted region and high-PV impacted data array. This process is completed by updating the LRU stacks associated with each cache block after swap operation.

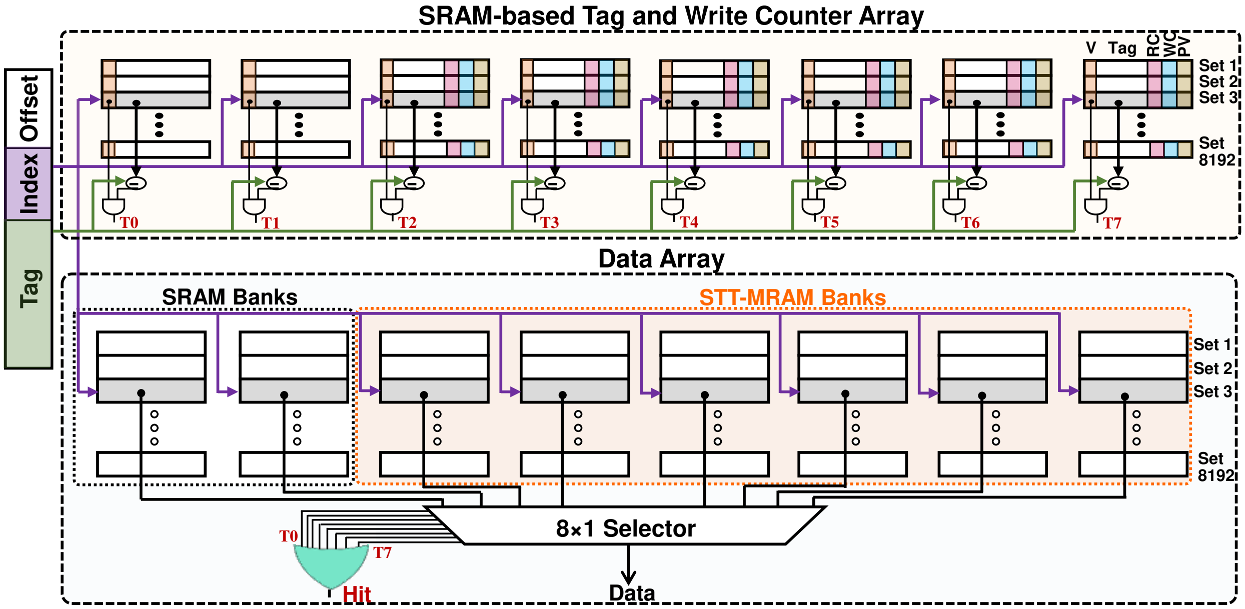


Figure 4.8: The scheme of hybrid 8-way set associative SRAM and STT-MRAM cache design, whereby each bank stores a way. In the above configuration, two SRAM-based banks and six STT-MRAM based banks are illustrated.

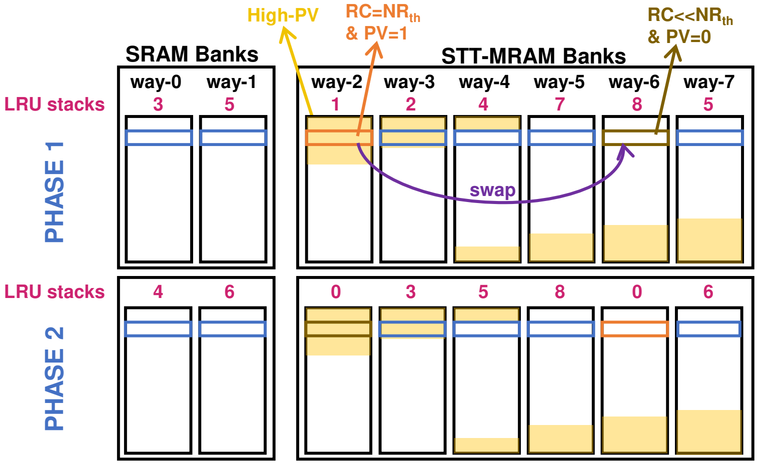


Figure 4.9: The migration policy to swap a read-intensive block resided in high-PV impacted region with not access intensive block located in low-PV impacted region.

Algorithm 2: Block Insertion/Migration Policy

```
Assumptions:
- RC: Read Counter, WC: Write Counter, PV: Process Variation Status
- $NR_{th}$: read threshold levell, $NW_{th}$: write threshold level
- Way 0-1 and Way 2-7 are built in SRAM and STT-MRAM (NVM), respectively in shared LLC
Function insertion() /*algorithm for inserting requested block*/
begin
    if LLC miss then
        if write miss then
            eviction() /*evict LRU block $\in$ $LLC_{SRAMBank}$*/
            copy block $\in$ memory into $LLC_{SRAMBank}$
        else if read miss then
            if $\exists$ block's address $\in$ read intensive block profiler then
                eviction() /*evict LRU block $\in$ NVM-$Bank_{PV=0}$*/
                copy block $\in$ memory into NVM-$Bank_{PV=0}$
            else
                eviction() /*evict LRU block $\in$ $LLC_{NVMBank}$*/
                copy block $\in$ memory into NVM Bank
    if read hit then
        if $\exists$ block's address $\in$ read intensive block profiler then
            update LRU status
        else if read intensive block profiler is full then
            evict LRU entry and fill the profiler with the new entry's address
        else if $RC_{block} < NR_{th}$ then
            ++$RC_{block}$
        else
            add new entry's address to read intensive block profiler
            migration()
    if write hit then
        if block $\in$ $LLC_{NVMBank}$ & $WC_{block} < NW_{th}$ then
            ++$WC_{block}$
        else if block $\in$ $LLC_{NVMBank}$ then
            migration()
Function migration()/*algorithm for migrate blocks*/
begin
    if read from block $\in$ $Bank_{PV=1}$ then
        swap (block $\in$ $Bank_{PV=1}$, block $\in$ ($Bank_{PV=0}$ & RC $< NR_{th}$))
    if write into block $\in$ $LLC_{NVMBank}$ then
        swap (block $\in$ $LLC_{NVMBank}$, block $\in$ $LLC_{SRAMBank}$)
```

4.3 Experimental Results

To comprehensively evaluate SOS efficacy, we analyze the SOS on both circuit- and architectural-level simulators. Experimental results are presented in Section 4.3.1 and 4.3.2 whereby the evaluation parameters are listed in Table 4.1.

4.3.1 Circuit-Level Simulation Results

Simulation results have been extracted using HSPICE based on the 22nm Predictive Technology Model (PTM) to calculate the power and performance of a 1-bit PCSA and SPCSA. The design parameters and PV values are listed in Table 4.1. Every design has been analyzed in an ideal case without PV, as well as Monte Carlo simulation in the presence of PV. The results for the analysis of ideal case are listed in Table 4.2.

Furthermore, 10,000 Monte Carlo simulations were performed considering different standard deviations for CMOS transistors' threshold voltage (V_{th}) and also MTJ's MgO thickness, shape, and area in order to cover the range of cases that may occur in the fabricated device. In particular, variation of 1% and 10% for the MTJs' resistance is assessed via Monte Carlo simulation. Based on the results listed in Table 4.2, it can be concluded that MSA operating in PCSA mode, outperforms MSA operating in SPCSA mode, however, it suffers more from PV. On the contrary, MSA operating in SPCSA mode offers improved performance in terms of reliability and PV immunity compared to MSA operating in PCSA mode due to its reduced BER.

Based on the results listed in Table 4.2, Table 4.3, and Table 4.4, it can be concluded that MSA operating in PCSA mode attains 6-fold improvement over MSA operating in SPCSA mode on average in terms of Energy Delay Product (EDP). This is due to an improvement of 2.43μW and 8.7ps on average offered in PCSA. On the contrary, MSA operating in SPCSA mode increases the relibility by having 6% reduced BER considering TMR=100% on average due to PV, compared to MSA operating in PCSA mode.

Furthermore, it is observed that by optimizing the reference MTJ and using (MTJ$_P$+MTJ$_{AP}$)|| (MTJ$_P$+MTJ$_{AP}$) configuration, the BER can be reduced by 15% on average (for TMR=100%). In addition, based on the results of Monte Carlo Simulation, it is clear that larger MTJ resistance, reduces the impact of variation on sensing output.

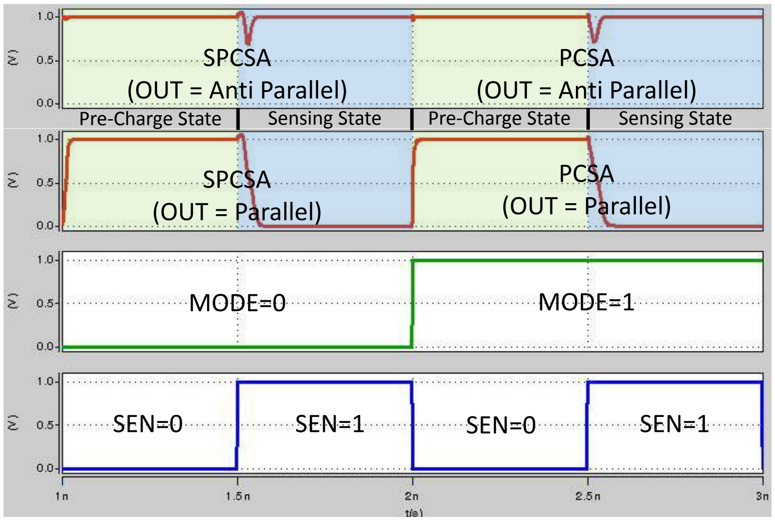


Figure 4.10: Transition waveforms of PCSA and SPCSA.

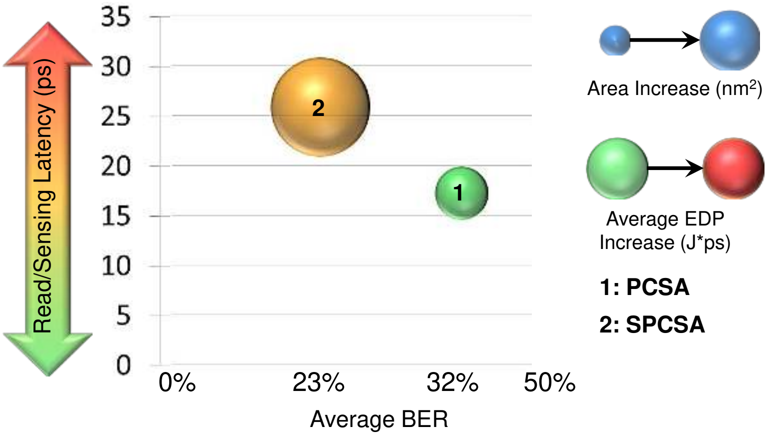


Figure 4.11: PCSA and SPCSA design space for TMR=100%.

Table 4.1: Evaluation parameters

Architectural Parameters

chip	4-core CMP
core	3.3GHz, Fetch/ Exec/ Commit width 4
L1	private, 32 KB, I/D separate, 8-way, 64 B, SRAM, WB
L2	shared, 4 MB, 8 banks, 8-way, 64 B, STT-MRAM, WB
memory	8 GB, 1 channel, 4 ranks/ channel, 8 bank/ rank

4MB L2 cache bank configuration (32nm, temperature=350K)

L2 Cache Technology	RL/WL $(cycles)$	RE (nJ)	WE (nJ)	LP (mW)
SRAM	7.43/5.78	0.161	0.156	295.58
STT-MRAM	9.08/25.58	0.216	0.839	18.39

SOS	9.08/25.58	PCSA=0.208
		SPCSA=0.218
			0.839	18.39

RL: Read Latency, WL: Write Latency, RE: Read Energy,
WE: Write Energy, LP: Leakage Power

Technology Parameters

Tech node	22nm
$pMOS$	$\mu(V_{th})$=460mV, $\sigma(V_{th})$=50mV (10%) $width(= 2 \times Length)$=44nm, $\sigma(width)$=0.44nm (1%)
$nMOS$	$\mu(V_{th})$=500mV, $\sigma(V_{th})$=50mV (10%) $width(= Length)$=22nm, $\sigma(width)$=0.22nm (1%)
MTJ	*The effects of variation are applied to TMR* MgO Thickness=0.85nm Shape Area(main MTJ)=(π/4)×40×40nm^2 Reference MTJ: Shape Area(MTJ$_{AP}$)=(π/4)×30×30nm^2 Shape Area((MTJ$_P$+MTJ$_{AP}$)$\|$(MTJ$_P$+MTJ$_{AP}$))= (π/4)×40×40nm^2 R.A(Resistance×Area)=5Ω.μm^2 TMR=100%, $\sigma(TMR)$=1% & 10% α(Damping Factor)=0.01 Nominal Voltage=1.0V, SEN Signal Period (T)=1ns

Table 4.2: Simulation Results for Ideal State (MTJ$_{Ref}$=5.7KΩ)

Design	**Area (Device Count)**			**Anti-Parallel (6.4KΩ)**			**Parallel (3.2KΩ)**		
	PMOS Trans.	NMOS Trans.	MTJ	Delay (ps)	Power (μW)	EDP (J*ps)	Delay (ps)	Power (μW)	EDP (J*ps)
PCSA	4	3	2	20.8	0.79	16.43	13.5	0.76	10.24
SPCSA	8	5	2	28.8	3.21	92.45	22.9	3.21	73.51

Table 4.3: MONTE CARLO SIMULATION 10,000 RUN RESULTS (MTJ$_{Ref}$=5.7KΩ, MTJ$_P$=3.2KΩ, and MTJ$_{AP}$=6.4KΩ for TMR=100%)

Design	Variation	BER (%) for TMR=						SBER($X \geq 16$)(%)	
	10% in V_{th}	**100%**	**150%**	**200%**	**250%**	**300%**	**350%**	**100%**	**150%**
PCSA	**1%** in TMR	38.29	17.15	5.89	1.65	0.30	0.03	11.96	0.002
SPCSA	**1%** in TMR	34.04	9.41	1.35	0.09	0.01	0.00	4.54	¡1E-6
PCSA	**10%** in TMR	38.44	17.18	6.06	1.62	0.34	0.04	12.32	0.002
SPCSA	**10%** in TMR	34.32	10.00	1.44	0.13	0.02	0.00	4.88	¡1E-6

Table 4.4: MONTE CARLO SIMULATION 10,000 RUN RESULTS (MTJ$_{Ref}$=4.8KΩ, MTJ$_P$=3.2KΩ, and MTJ$_{AP}$=6.4KΩ for TMR=100%)

Design	Variation	BER (%) for TMR=						SBER($X \geq 16$)(%)	
	10% in V_{th}	**100%**	**150%**	**200%**	**250%**	**300%**	**350%**	**100%**	**150%**
PCSA	**1%** in TMR	24.87	9.01	2.71	0.54	0.05	0.00	0.18	¡1E-6
SPCSA	**1%** in TMR	17.68	3.41	0.32	0.02	0.00	0.00	0.003	¡1E-6
PCSA	**10%** in TMR	24.90	9.31	2.69	0.59	0.07	0.00	0.19	¡1E-6
SPCSA	**10%** in TMR	17.78	3.58	0.35	0.02	0.00	0.00	0.003	¡1E-6

4.3.2 Architecture-Level Simulation Results

The latency and energy usage associated with read and write operations for SRAM and conventional SA cache accesses are provided by NVSim [41]. However, we integrate the obtained results from Section 4.3.1 for 1-bit PCSA/SPCSA into NVSim to extract the power and performance parameters for the cache accesses in SOS design. PARSEC 2.1 benchmarks suite executed on modified MARSSx86 [130] which supports asymmetric cache read and write from distinct cache banks to extract the evaluation parameters of different cache designs during program execution. We model a Chip Multi-Processor (CMP) with four single-threaded x86 cores. Each core consists of private L1 cache, and shared LLC among all the cores. Eleven workloads are executed for 500 million instructions starting at the Region Of Interest (ROI) after warming up the cache for 5 million instructions. The `simsmall` input sets are used for all PARSEC workloads.

4.3.2.1 Energy Usage Comparison

In order to evaluate the energy benefit of SOS, we compare the energy breakdown of SOS MSA/ASA with LLC built upon SRAM, STT-MRAM, and Hybrid Cache enabled SOS (HC-SOS) with migration policy. Based on the extracted results from NVSim which are listed in Table 4.1, SOS neutralizes the high energy consumption of SPCSA/VISA via low-power PCSA/EASA during read operation. As illustrated in Fig. Fig. 4.12, the high write energy overhead for storing a value into STT-MRAM cell incurs significant energy overhead in both SOS MSA/ASA and STT-MRAM based LLC while the SRAM-based LLC design benefits from symmetric acceptable energy consumption for both read and write operations. This incident is conspicuous for write-intensive workloads such as **`facesim`**, **`ferret`**, and **`vips`** where the ratio of write accesses to the LLC is significantly more than read accesses. We address this issue by proposing HC-SOS where SRAM banks are considered to accommodate write-intensive blocks while read-intensive blocks are maintained in low-PV impacted regions of STT-MRAM. The experimental results indicate that HC-SOS can save up to 10.6% on average of dynamic energy consumption compared to STT-MRAM based LLC design. Although SRAM exhibits lower dynamic energy consumption, its high leakage power has exacerbated the overall consumed energy compared to other designs as shown in Fig. 4.13. Both STT-MRAM and SOS MSA/ASA can conserve 88% on average of the total consumed energy. HC-SOS incurs higher leakage energy compared to STT-MRAM and SOS MSA/ASA due to leveraging two SRAM-based banks in the design incurring relatively more leakage energy to the entire cache subsystem.

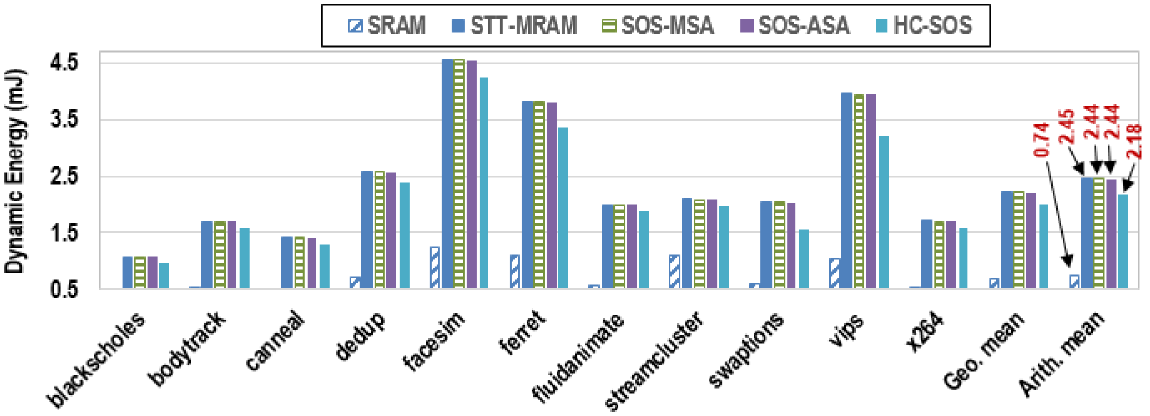


Figure 4.12: LLC dynamic energy comparison for SRAM, STT-MRAM, SOS-MSA, SOS-ASA, and HC-SOS, respectively.

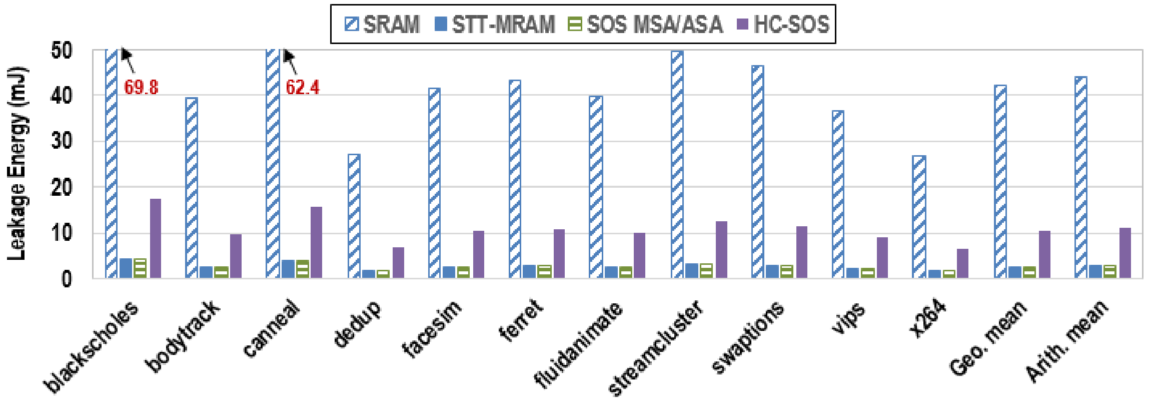


Figure 4.13: LLC leakage energy comparison for SRAM, STT-MRAM, SOS-MSA, SOS-ASA, and HC-SOS, respectively.

4.3.2.2 Write Performance Analysis

SRAM-based LLC exhibits greater write performance compared to regular STT-MRAM, SOS, and HC-SOS. The main reason for performance degradation in STT-MRAM and SOS designs is the high write latency while this latency has been amortized in HC-SOS. HC-SOS reduces the latency associated with write operation via allocating write-intensive cache blocks from SRAM ways for faster write response which in turn results in improved performance. On the other hand, HC-SOS leverages STT-MRAM to maintain read-intensive blocks for a long duration without sacrificing significant energy for preserving data. Fig. 4.14 illustrates the cumulative LLC write latency during workload execution whereby HC-SOS improves the write performance by 12.4% on average compared to STT-MRAM design. The experimental results show that the workloads, such as `vips`, `swaptions` and `ferret`, leverage the full potential of HC-SOS to further diminish the high write latency which adversely impacts the entire cache subsystem throughput and accessibility.

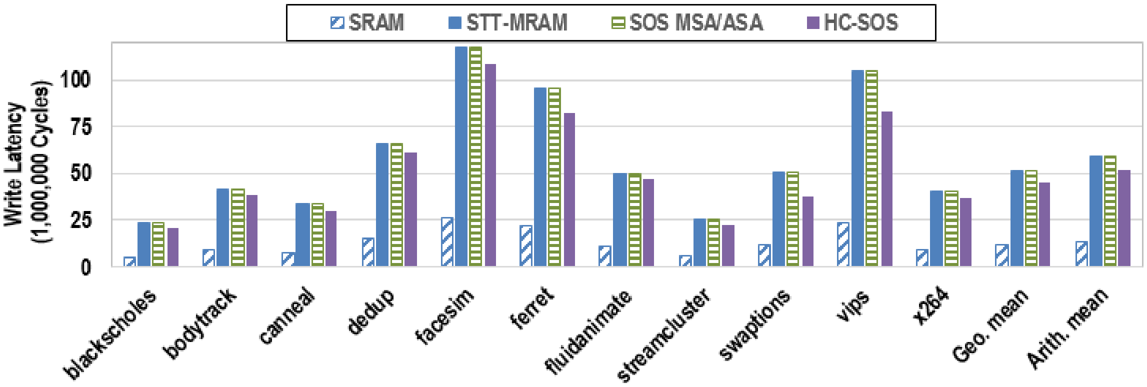


Figure 4.14: Write performance comparison for SRAM, STT-MRAM, SOS MSA/ASA, and HC-SOS.

4.3.2.3 Empirical Fault Model Analysis

Fig. 4.15 illustrates the comparison of distribution of sensed data between LLC built by STT-MRAM, SOS utilizing MSA design, SOS leveraging ASA design, and HC-SOS-ASA design. We assume that the PV map for each cache bank is similar to the floorplan of STT-MRAM layer, shown in Fig. 4.2. We apply the PV ratio of each accessed sub-bank during fault analysis for each workload. For example, if a sub-bank experiences a high amount of PV, it is highly likely that the data will be sensed incorrectly. Our experimental results indicate that the PV effect may incur around 27.5% of the sensed data to be read incorrectly from which 21.5% are extremely vulnerable which implies that around one fifth of the overall sensing operations have the potential to contaminate the application's data structure. If this rate of sensed data are not accommodated properly, it may induce application crashes or prolong the program execution. Across all benchmarks suite, the calculated VFDS for some benchmarks is more than others. For example, in **`blackscholes`** and **`canneal`** workloads, the proportion of read operations and dirty victim blocks residing in LLC are more than write operations which results in the increased VFDS. As another example, the **`streamcluster`** workload is a read-intensive application in which more than 85% of memory operations are read accesses which increase the chance for enduring higher VFDS. On the other hand, SOS addresses the probability of sensing incorrect data through leveraging PV-resilient SAs array in the sub-bank architecture whenever the sub-bank's PV ratio is more than a predefined threshold. The proposed PV/energy aware cache block migration improves further the SOS throughput by relocating read/write intensive blocks which results in enhanced TDS, write performance, and bank service time. Namely, the VFDS in the HC-SOS-ASA design is reduced by 89% on average compared to LLC with STT-MRAM, thus improving the mean TDS from 72.5% to 97% across all workloads.

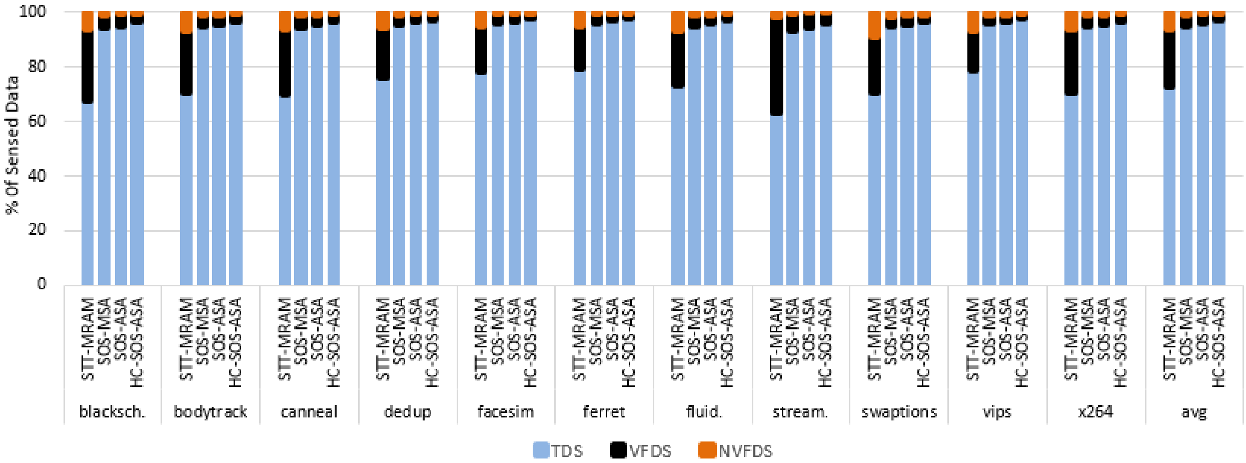


Figure 4.15: Distribution of sensed data. SOS is equipped with MTJ_{Ref}=4.8KΩ, TMR=150%, and 10% variation in TMR.

4.4 Conclusions

In conclusion, SOS is a circuit-architecture cross-layer solution which combats the common PV problem in the emerging NVM technologies by engaging PV-resilient SAs array offering acceptable resistive sensing margin. In addition, SOS manages to meet the constrain power budget in IoT devices through low-power SAs in sub-banks that experience lower rates of PV. Furthermore, we classify the output of the sensed data based on its impact on the execution flow of the workload. This classification of experimental outcomes is vital to identify the efficiency of SOS to accommodate critical read operations. Our experimental results indicate that the energy consumption of SOS is as high as LLC with conventional SAs, while SOS reduces the overall VFDS significantly. The confluence of these factors in turn significantly increase the reliability of realistic program execution. Furthermore, our results exhibit that SOS-enabled hybrid cache improves the write performance by 12.4% on average compared to STT-MRAM design. On the other hand, the VFDS is reduced by 89% on average in the SOS-enabled hybrid cache compared to LLC with STT-MRAM. This improves the mean TDS from 72.5% to 97% across all workloads.

CHAPTER 5: ACCELERATE SERVICE TO CRITICAL LOADS

To combat the deficiencies of SRAM, emerging device technologies such as PCM, RRAM, and STT-MRAM have been introduced as alternative solutions [123, 167]. To maximize the benefit of extra capacity realized by replacing SRAM with emerging candidates, various hybrid cache designs are utilized to maintain the read-intensive cache blocks in the high-retention region while write-intensive blocks are allocated to low-retention regions of cache [161, 81]. In this work, we adopt the same strategy for prioritizing the service to 1) Extensive Read Reused Access (ERRA) blocks that remain unchanged during their residency in LLC, and 2) frequently reused blocks which may exhibit distant re-reference intervals in L2. An efficient block allocation/replacement policy is introduced herein to maximize the throughput of the adopted hybrid cache design. To minimize the overhead of finding and labeling the potential blocks that must be prioritized to copy into high-retention regions of the adopted hybrid L2 cache, we propose a novel Multi-level Access History Profiler (MAHP) inspired by observing the high spatial locality at page-level granularity among different class of workloads. To be specific, we observed that more than 50% of read accesses are from one of the three pages with the most accesses in the selected workloads, indicating that the majority of cache block accesses are from the same physical page. MAHP aggregates the LLC block addresses tagged with identical Most Significant Bits (MSBs) into a single entry while their remaining Least Significant Bits (LSBs) of the addresses are stored into distinct LSB-address entries. Compared to the existing works, this paper provides the following contributions:

- new insights regarding the distribution of read reused cache blocks, and the spatial locality rate at a page-level granularity resulting in the MAHP strategy,
- a low-conflict in-situ monitoring mechanism to track LLC traffic and autonomously copy block candidates to the upper-level High Retention STT-MRAM Cache (HRSC) of each

processor core,

- improve the overall read service time by accelerating the service to the critical loads versus stores,
- identify preferred designs for Low Retention STT-MRAM Cache (LRSC) by comprehensively evaluating the candidates in terms of energy consumption and performance delivery,
- optimization strategies for balancing workload lifetime performance versus energy reduction from diminished write latency, and
- propose *Reclusive*, an efficient block allocation/replacement policy to maximize throughput of the adopted hybrid cache.

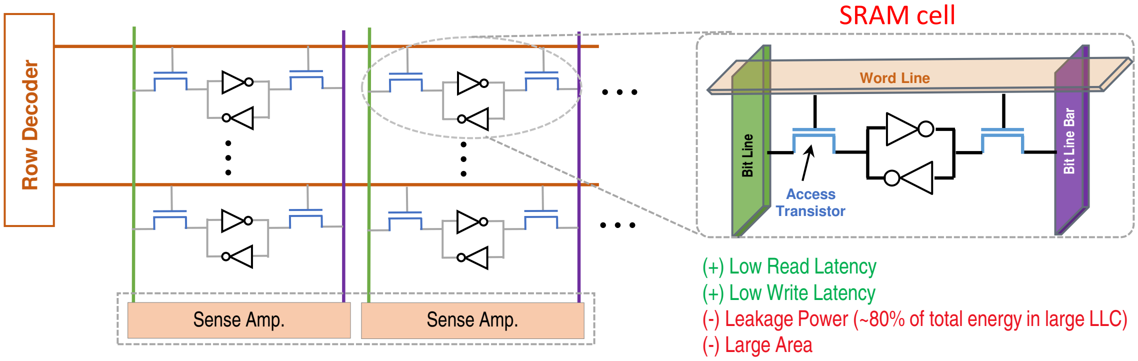
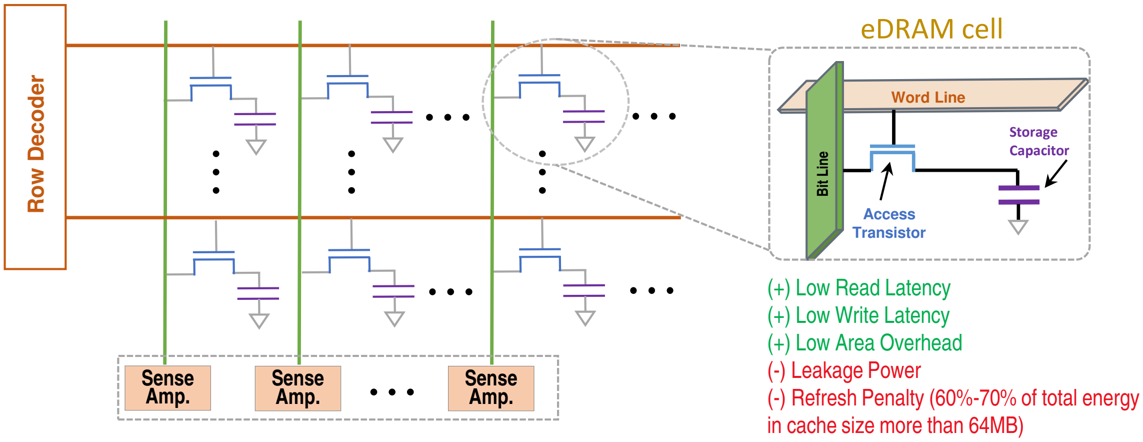
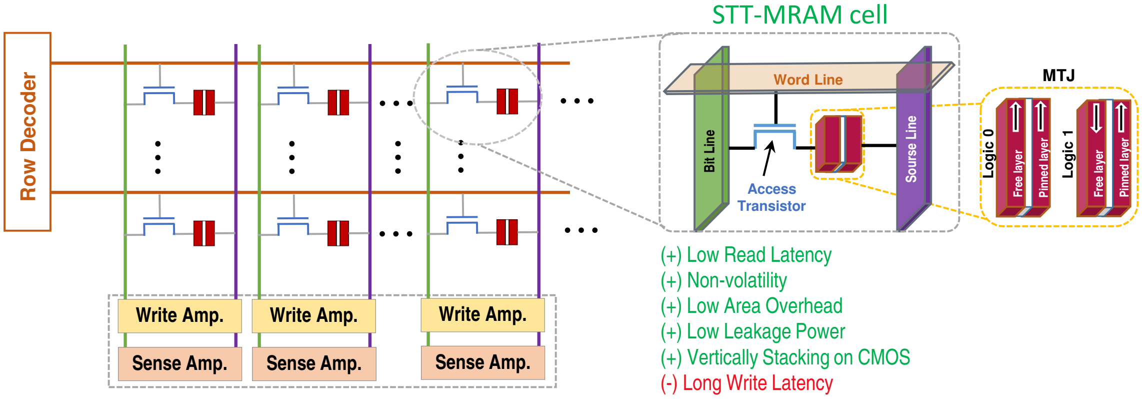


Figure 5.1: LLC organization based on SRAM.



(+) Low Read Latency
(+) Low Write Latency
(+) Low Area Overhead
(-) Leakage Power
(-) Refresh Penalty (60%-70% of total energy in cache size more than 64MB)

Figure 5.2: LLC organization based on eDRAM.



(+) Low Read Latency
(+) Non-volatility
(+) Low Area Overhead
(+) Low Leakage Power
(+) Vertically Stacking on CMOS
(-) Long Write Latency

Figure 5.3: LLC organization based on STT-RAM.

5.1 Background on Technology Trends for Cache Organization

The utilization of SRAM in favor of the performance improvement incurs the cost of increased energy consumption and high area overhead. To address this issue, research interest is increasing for alternative solutions which offer the higher energy-efficient, cost-effective, large-capacity, and competitive read/write operation speed compared to the traditional SRAM arrays as illustrated in Fig. 5.1. Embedded DRAM (eDRAM) is one of the promising solutions pioneered in the design of on-chip memory hierarchy. As depicted in Fig. 5.2, the eDRAM cell consists of a storage capacitor which is connected to the *bitline* wire through the access transistor. Even though the employment of a capacitor for maintaining the logic value has significantly saved area compared to SRAM that utilizes 6 transistors per bit cell, the eDRAM technology suffers from high dynamic energy consumption due to mandatory periodic refresh required to keep the stored value in the valid state [166]. It has been reported in [5] that the refresh scheme contributes around 70% to the overall energy consumption in LLC. Furthermore, the periodic refresh operation limits the CPU accesses to the eDRAM arrays which leads to typically deployment of the eDRAM as LLC by the memory designers.

Another promising alternative solution to replace SRAM is STT-MRAM. Non-volatility and near-zero leakage power are two prominent characteristics of STT-MRAM cells. Furthermore, STT-MRAM offers at least 3x to 4x area savings compared to SRAM [147]. As illustrated in Fig. 5.3, STT-MRAM cache leverages an extra device to drive large current for a certain period when the write operation is issued. The high write current and slow write operation of STT-MRAM has motivated researchers to devise novel architectures to overcome these new challenges while developing STT-MRAM based L2 between L1 and LLC [28, 161, 104, 97, 177, 121, 108]. Herein, we address all of the above issues through adjustment of the intrinsic non-volatility characteristics of STT-MRAM to achieve SRAM-competitive performance while incurring low energy consumption.

5.2 Motivation

We classified the motivation behind our technique into two subsections. The main reason is that our observation verified that these incidents have cause and effect relationship which can be explored from two perspectives of *memory reference to shared LLC* and *clean victim blocks eviction in private L2*.

5.2.1 Analysis of Memory Access Pattern to Shared LLC

The cache lines, which are brought into a shared LLC, can be classified into two categories, namely, *non-reused* and *reused* cache lines. A non-reused cache line in LLC does not experience more than one read access, while a reused cache line is accessed from multiple cores during its residency in the LLC. Fig. 5.4 illustrates the distribution of read accesses to the cache line, in which the majority of the LLC lines are non-reused cache lines, as indicated by the leftmost blue column for each benchmark which averages 50.1% across the benchmark suite [1].

Although the majority of the LLC line read accesses are non-reused cache lines, this portion of cache space is accessed only once for read operation purpose during program execution which results in spending a small fraction of time to read from the non-reused cache lines. On the other hand, reused cache lines with high read access rate, e.g. read more than 64 times, have experienced a significant amount of program read operations, e.g. 16.2% on average, while they only occupy 1.55% of the entire cache space as depicted in Fig. 5.4 and 5.5.

These reused cache lines can experience either *read-only access* or *multifaceted access,* e.g. write, insert, evict. The exclusive-read reused cache lines, whose read accesses are more than 64, account for 8.37% of the total read accesses as shown in Fig. 5.6. Herein, we refer to this group of cache lines as Extensive Read Reused Access (ERRA) blocks that remain unchanged during program runtime. This observation motivated us to re-design the conventional cache design in favor of ERRA cache blocks to reduce the response time to read requests from LLC which in turn cause overall IPC improvement. Moreover, our goal align with the recent works that attempt to prioritize the service to performance critical read reused cache blocks [89, 134, 151] for achieving a better performance.

[1] The experimental setup is explained in Section 5.4

The ERRA blocks are excellent choices to be stored into high retention STT-RAM arrays because 1) these lines are written once while experiencing read-only operations for the remainder of program execution, 2) the energy and performance overheads for accessing ERRA blocks are small because once the ERRA block is brought to the high retention STT-RAM array, it only will be accessed by the read operations prior to its eviction.

The characteristics of high versus low retention memory arrays are identified in the following Section.

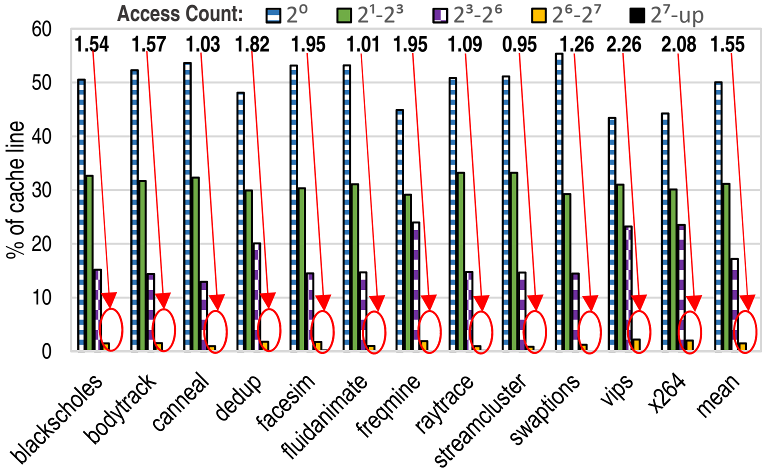


Figure 5.4: Percentage of total cache lines experiencing read operation.

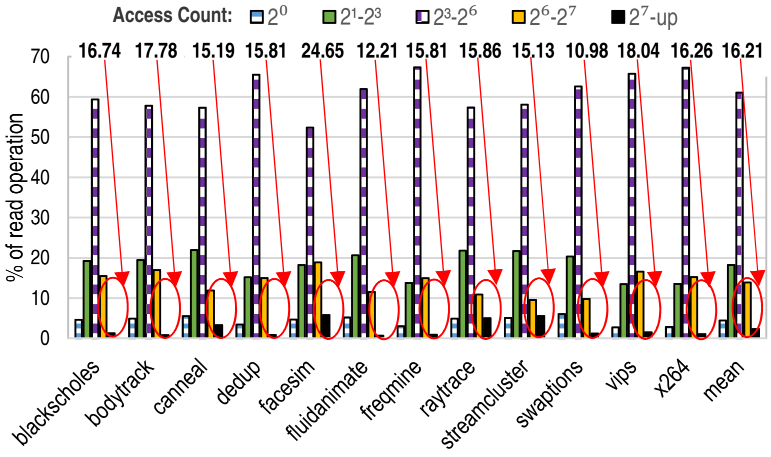


Figure 5.5: Proportion of read reference activity.

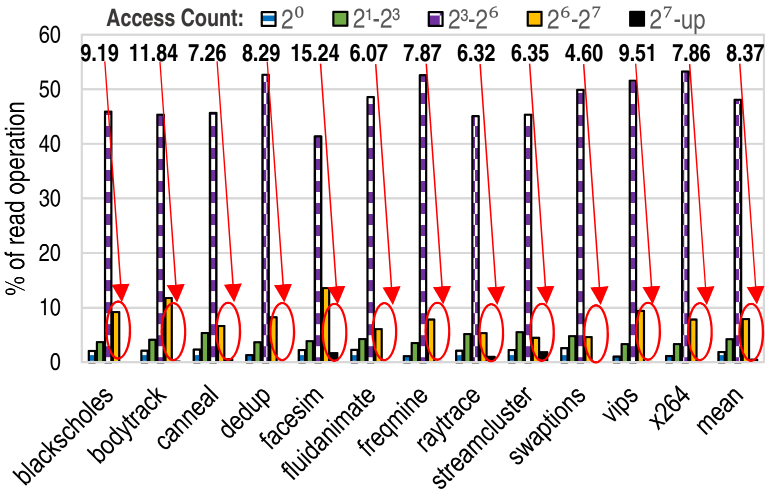


Figure 5.6: Proportion of exclusive-read reference activity.

5.2.2 Analysis of Clean Victims Eviction in Private L2

Each LLC access (either hit or miss) is followed by finding the victim block in L2 to be replaced by the accessed LLC block. Based on non-inclusive property [174], the clean victim blocks are silently dropped from L2 during replacement process while the dirty victims are inserted into their replica blocks in LLC. Yet, the replica block may not exist in LLC for non-inclusive design which necessitates 1) eviction of a victim block from LLC, and 2) insertion of dirty victim from L2 into LLC.

In our study, we focus on improving the temporal locality of a fraction of working set which contributes significant read hits to L2. To be specific, we investigate the reference pattern to the blocks whose their residency in L2 reduce the L2 read miss ratio over time. However, these blocks are evicted from L2 before contributing acceptable L2 read hits due to the weak temporal locality or the ineffective replacement policy or the insufficient capacity to retain the referenced fraction of working set [138]. The consequence of this effect is that whenever there is a read miss on aforementioned blocks in L2, a significant delay and energy consumption would be imposed to the system to bring the referenced block from LLC or memory to L2 while this block may not be re-referenced again during its residency in L2. Fig. 5.7 illustrates the proportion of the frequently re-referenced clean victim blocks after their eviction from L2 for the baseline 512KB 8-way L2 cache. For example, around 20.4% of the evicted clean victim blocks from L2 are re-referenced more than 16 times on average during the program execution. This means that whenever the processor calls for one of the evicted clean victim blocks from L2, it must stall until the requested block be transferred from either LLC or memory to L2. If we add up the processor stalls for the frequently re-referenced clean victim blocks, it incurs significant energy consumption and delay overhead which increasingly sabotage the throughput of the system and aggravate the considered budget for the energy.

The RRAP tackles this problem through the long-enough accommodation of highly read accessed fraction of working set in L2 to accelerate the service to these critical loads and improve the read hits to L2.

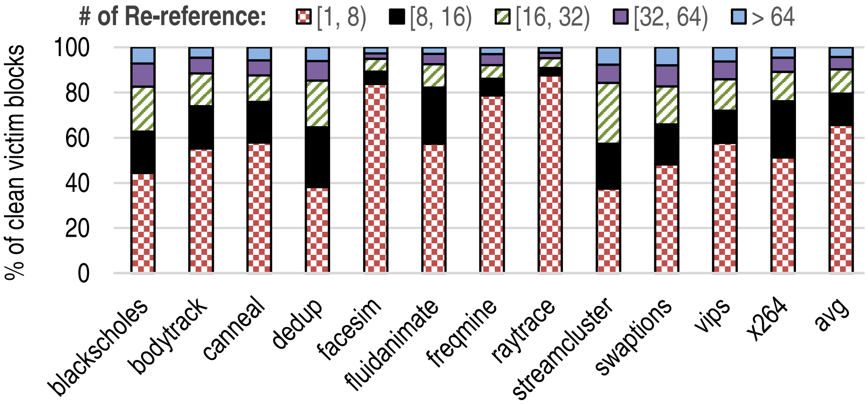


Figure 5.7: Proportion of the frequently re-referenced clean victim blocks after their eviction from L2.

5.3 Technical Approach

The intuition behind the adopted hybrid L2 cache design is to benefit from the extra capacity provided by replacing SRAM-based L2 with STT-MRAM to accommodate the replica of ERRA cache lines from LLC and the frequently evicted clean blocks from L2. The proposed schematic view is illustrated in Fig. 5.8 and referred to Read Reference Activity Persistent (RRAP) cache design, in which the service to frequently-read cache lines of LLC is accelerated while energy consumption is significantly amortized due to the near-zero standby power property of STT-MRAM arrays. To achieve this goal, the Reclusive cache allocation policy, inspired by FLEXclusive[154], is adopted in RRAP which will be presented in the following section.

5.3.1 Read Reference Activity Persistent (RRAP) Cache Hierarchy

Hybrid cache designs improve write performance while offering much larger cache capacities with low leakage power [160, 166, 161, 81]. The RRAP strategy has been inspired by these techniques, but realizes a novel allocation/replacement policy to substantially reduce the service time of read-intensive workloads. The baseline configuration of RRAP consists of a hybrid 8-way set associative LRSC and HRSC design, where way-0 through way-3 are implemented within LRSC banks while way-4 through way-7 constitute HRSC banks. This configuration is selected based on our experimental results, whereby the four blocks in each LRSC set were adequate to maintain an acceptable performance before/after placing the ERRA and the frequently evicted clean blocks into the HRSC region. Since the peripherals required for read and write operations in NVM arrays occupy a larger portion of the cache footprint than peripherals required by SRAM arrays, it is beneficial to build the tag array with SRAM cells. Thus, we assume that the entire tag array is built with SRAM [124].

Unlike conventional cache design approaches, where the tag and data array are accessed simultaneously to reduce access latency while incurring significant power overhead, starting with the beneficial concept to split the cache access into two stages similar to the work presented in [166], we refocus on extensions to favor high RRAP throughput. If L2 is accessed with a read operation, the tag array and all LRSC/HRSC banks are accessed in parallel to look for the data block in both regions of L2. This aligns with RRAP goal for reducing service time to the read request. Even though the parallel search for finding the data block incurs more energy consumption due to enabling two exclusive resources, the gained performance will be shown to compensate this overhead. The energy consumption overhead for these resources has been considered in our simulation results.

If the cache set is accessed by a write operation, the tag arrays and LRSC banks are searched during the first stage. If the data is not found in LRSC banks, the unnecessary accesses to HRSC banks can be skipped while LLC banks are accessed in the next stage. Our proposed allocation/replacement policy maintains the read-dominant cache blocks in HRSC banks while write-intensive and non-access-intensive blocks are transferred to LRSC banks.

The ratio of read and write accesses to LLC blocks are monitored through a lightweight unit that will be presented in Section 5.3.3. RRAP promotes LLC blocks to either LRSC or HRSC based on the history of accesses. The promotion to HRSC only occurs if a LLC block experiences frequent read accesses. We conducted an extensive exploration to attain the preferred value for the read threshold level, NR_{th}, within our design. We found that when NR_{th} is small, the ratio of blocks that must be copied into HRSC significantly increases. However, note that the limited capacity of HRSC constrains the number of read-intensive cache blocks that can be maintained. This results in a high ratio of cache blocks to be frequently replaced without providing adequate read services which in turn incurs significant write overhead and undermines the read-friendly property of HRSC. On the other hand, if NR_{th} is too large, then the HRSC utilization significantly decreases because only a few read-intensive cache blocks are selected to be brought into HRSC. Based on our experimental results, NR_{th} equal to *64 read accesses to the LLC block*, maximizes the HRSC utilization while incurring an acceptable cache block replacement rate in HRSC across a wide range of workloads, as discussed herein.

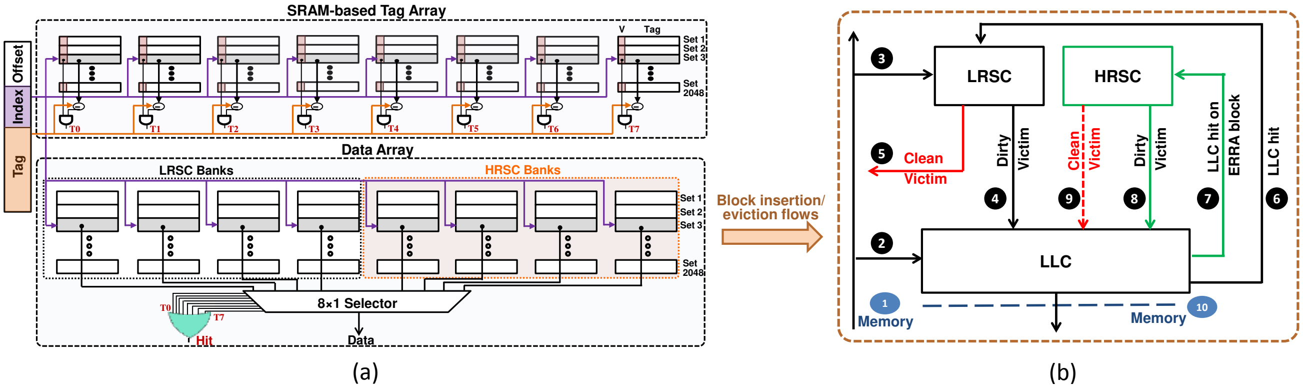


Figure 5.8: RRAP scheme: (a) Heterogeneous split cache architecture, (b) Block insertion/eviction flows.

5.3.2 RRAPclusive: Efficient Cache Configuration to Maximize RRAP Throughput

BBesides considering the energy-efficient emerging devices in upper-level cache to accelerate service to the critical loads through increased cache capacity and reduced energy consumption, we also propose an efficient block allocation/replacement policy called *Reclusive*, a cache policy redesign of exclusive and non-inclusive designs, to maximize the throughput of hybrid cache designs entailing low and high retention STT-MRAM arrays. The block allocation/replacement flow in Reclusive is illustrated in Figure 5.8 (b) and its corresponding algorithm is presented in Algorithm 1. On an LLC miss, the requested block is brought into LRSC and LLC. Transferring the requested block from memory to cache incurs one off-chip (①) and one on-chip (②,③) traffic.

Upon an LLC write hit, the requested block is brought into LRSC while the replica remains in the LLC (⑥). Likewise, the cache blocks are brought into LRSC on an LLC read hit while the LLC still keeps a duplicate copy of block (⑥), unless the cache block belongs to the ERRA group experiencing zero update memory operation, whereby the HRSC design is chosen for copying the requested block (⑦).

Upon an LLC access, the victim block in L2 must be selected for replacement by the accessed

LLC block. Based on the history of access patterns to a LLC block, Reclusive selects the victim blocks either from LRSC or HRSC. The block eviction for LRSC is similar to the inclusive and non-inclusive caches, whereby the clean victim blocks are silently dropped (⑤) while the dirty victims are inserted into their replica blocks in LLC (④). Yet, the replica block may not exist in LLC for non-inclusive designs which necessitate the eviction of a victim block from LLC. If the evicted victim block from LLC is dirty, the block must be written back into the main memory which results in an off-chip transfer (⑩).

On the other hand, the employment of the non-inclusive approach to deal with the victim blocks of HRSC incurs a significant amount of energy and delay cost to retrieve the re-referenced block. To overcome this challenge, we propose the Reclusive strategy to exploit the exclusive feature to deal with the victim blocks from HRSC, whereby both clean and dirty blocks must always be inserted into LLC (⑧ or ⑨) regardless of their status in HRSC. In addition, we specify Reclusive in such way to immediately promote the evicted clean victim blocks from HRSC as ERRA blocks during the insertion into LLC. This consideration aids to place the recently evicted clean blocks into HRSC cache again, if they are reused. Hence, it amortizes the corresponding service time to the critical loads and mitigates cache thrashing.

5.3.3 Multi-level Access History Profiler

Finding and labeling the ERRA blocks in LLC is on the critical path of data reference, which implies that the look up process must be completed quickly while incurring only a slight overhead on cache management and on energy consumption. An intuitive approach is to embed an additional 6-bit RC and 1-bit WC in each tag unit associated with the cache sets. Even though this approach may seem straightforward to implement, the large area overhead that is incurred undermines its simple design. In particular, RRAP only labels around 1.5% of LLC cache blocks as ERRA while

Algorithm 3: RRAPclusive Block Insertion/Eviction Policy

```
Assumptions:
- RC: Read Counter, WC: Write Counter
- $NR_{th}$: read threshold level
- LRSC and HRSC are private non-LLC in core i where $i = \{1, 2, ..., number\ of\ cores\}$, L3 is shared LLC
Function insertion() /*algorithm for inserting requested block*/
begin
    if LLC miss from core i then
        eviction() /*evict victim block $\in$ LRSC$_{core\ i}$ & LLC*/
        copy block $\in$ memory into LRSC$_{core\ i}$ & LLC
    else
        if LLC write hit from core i then
            eviction() /*evict victim block $\in$ LRSC$_{core\ i}$*/
            copy block $\in$ LLC into LRSC$_{core\ i}$
            $WC_{LLC\ block}$ = 1
        else if LLC read hit from core i then
            if RC == $NR_{th}$ and WC == 0 then
                if HRSC$_{core\ i}$ is full then
                    eviction() /*evict victim block $\in$ HRSC$_{core\ i}$*/
                copy $block \in LLC$ into HRSC$_{core\ i}$
            else
                eviction() /*evict victim block $\in$ LRSC$_{core\ i}$*/
                copy block $\in$ LLC into LRSC$_{core\ i}$
                ++$RC_{LLC\ block}$
Function eviction()/*algorithm for evicting victim block*/
if ($\exists$ dirty victim block $\in$ LRSC$_{core\ i}$/HRSC$_{core\ i}$) OR ($\exists$ clean victim block $\in$ HRSC$_{core\ i}$) then
    if $\exists$ replica block $\in$ LLC then
        update replica block $\in$ LLC
        /*insert dirty victim into same position in LLC,
        if $\exists$ clean victim block $\in$ HRSC$_{core\ i}$ => no LLC update is needed*/
    else if ($\nexists$ replica block $\in$ LLC) AND ($\exists$ dirty victim block $\in$ LRSC$_{core\ i}$/HRSC$_{core\ i}$) then
        evict victim block $\in$ LLC into memory
        insert (dirty victim block $\in$ LRSC$_{core\ i}$/HRSC$_{core\ i}$) into LLC
        set $WC_{inserted\ block\ \in\ LLC}$ = 1
        set $RC_{inserted\ block\ \in\ LLC}$ = 0
    else if ($\nexists$ replica block $\in$ LLC) AND ($\exists$ clean victim block $\in$ HRSC$_{core\ i}$) then
        evict victim block $\in$ LLC into memory
        insert (clean victim block $\in$ HRSC$_{core\ i}$) into LLC
        set $RC_{inserted\ block\ \in\ LLC}$ = $NR_{th}$
else if $\exists$ clean victim block $\in$ LRSC$_{core\ i}/LLC$ then
    silently drop clean victim block
else if $\exists$ dirty victim block $\in$ LLC then
    evict victim block $\in$ LLC into main memory
```

the remaining cache blocks remain unclassified, highlighting the expense of such a rudimentary profiling strategy.

Another approach that can be leveraged to identify the ERRA blocks is to employ a small read counter in the tag unit to determine the potential blocks that may experience frequent read accesses in the future. If the read counter reaches its threshold, then the address of the LLC block is copied into a single entry of a buffer that has been equipped with larger read counter for counting the remaining read accesses to the block. However, this design also incurs significant overhead to the memory subsystem since at least 32 bits must be stored in each single entry as the address of each LLC block. Considering the fact that many LLC blocks must be tracked to determine ERRA blocks, such a design also becomes undesirable.

In this regard, we show that the access patterns for a majority of workloads exhibit substantial spatial locality at a page-level granularity. This means that the majority of LLC blocks have identical physical tags. Based on this observation, we propose a low-area overhead Multi-level Access History Profiler (MAHP) to efficiently identify ERRA blocks. This tracking strategy for recording the frequently read LLC blocks incurs significantly less overhead compared to a default encoding. The idea is similar to the approach of using the small and large counter in a hierarchical approach, while the address of the targeted LLC block is stored in a multi-stage fashion. In particular, the address of LLC blocks that show the potential to become ERRA blocks are stored using MAHP, instead of employing large read/write counters in each tag associated with the sets of LLC. Accordingly, significant area overhead is eliminated.

5.3.3.1 Spatial Locality at Page-Level Granularity

In contemporary memory subsystems, the virtually-indexed physically-tagged approach is typically used to look up entries within cache, whereby the virtual index is extracted from the virtual address to find the block in the cache and to take out the physical tag. To verify that the proper block is selected, the physical tag is compared with the physical page address. Translation Lookaside Buffer (TLB) is used to map the virtual addresses to the physical page addresses. Thus, in workloads with high spatial locality, we observe that the majority of cache block accesses are from a single physical page, which means that these blocks have identical physical tags. Fig. 5.9 illustrates the fraction of read accesses for the top three pages with the most accesses in five selected workloads from PARSEC benchmarks. Approximately 97% of read accesses in `blackscholes` are from physical page B, indicating that the physical tag is identical for cache blocks that are fetched from this physical page. The same access pattern is seen in benchmarks `bodytack`, `facesim`, and `swaptions` where over 50% of read accesses are from one of the top three pages with the most accesses. On the other hand, the bar plots on the leftmost column of `dedup` show a symmetric access distribution among pages A, B, and C that indicates the spatial locality is evenly distributed among aforementioned physical pages for this workload. We also observed that 3% of read accesses on average share a similar 16-bits in the MSBs of their physical address in page B of `blackscholes` workload. This ratio of similarity among 16 MSBs addresses of LLC blocks decreases in benchmarks `bodytack`, `facesim`, and `swaptions`, but still more than 0.5% of LLC read accesses are similar in their 16 MSBs of their physical addresses.

5.3.3.2 Implementation of Multi-level Access History Profiler

These observations have motivated us to propose MAHP, in which the identical 16 MSBs of the physical address of ERRA blocks are stored in a single MSB-address entry corresponding to the blocks that have different values in their remaining Least Significant Bits (LSBs) of address, i.e. herein referred to as LSB-address entries. We consider a 6-level RC for each LSB-address entry to keep track of the blocks that experience frequent read accesses. This design eliminates the storing of repetitive MSBs addresses, which leads to significant reduction of the design and power consumption overhead. Fig. 5.10 illustrates the clear-box view of a feasible approach for integrating MAHP into the memory subsystem. This design can be realized by the ultra low-power TCAM unit proposed in [71] whereby the search process is split into multiple stages. Next, the selective TCAM rows at each stage are activated based on hits of previous stages.

Algorithm 2 demonstrates the scheme of storing the address of LLC blocks that have the potential to become ERRA in the future. During an LLC read hit, the 2-bit RC of the accessed block in LLC is incremented. If the RC resided in the tag store associated with LLC block reaches its threshold, the LLC block address is copied into MAHP. In order to copy the address of LLC block to MAHP, the following steps are required:

- search in MAHP to check if similar 16 MSBs pattern is available as an MSB-address entry,
- if the entry is not found, an empty MSB-address entry is allocated to store the 16 MSBs of LLC block address,
- an empty LSB-address entry is allocated and associated to its MSB-address entry to store 16 LSBs of LLC block address,
- the 6-bit RC of LSB-address entry is incremented by each read access to LLC blocks having an address in the MAHP,
- if the RC of LSB-address entry for accessed LLC block reaches the threshold value, then the block must be copied into HRSC.

We conducted a sensitivity analysis on different class of workloads such as computer vision and image processing, numerical computation, data compression, and data mining, to quantify the optimal number of MSB/LSB addresse entries. The outcome of this study will be presented in Section 5.4.2.

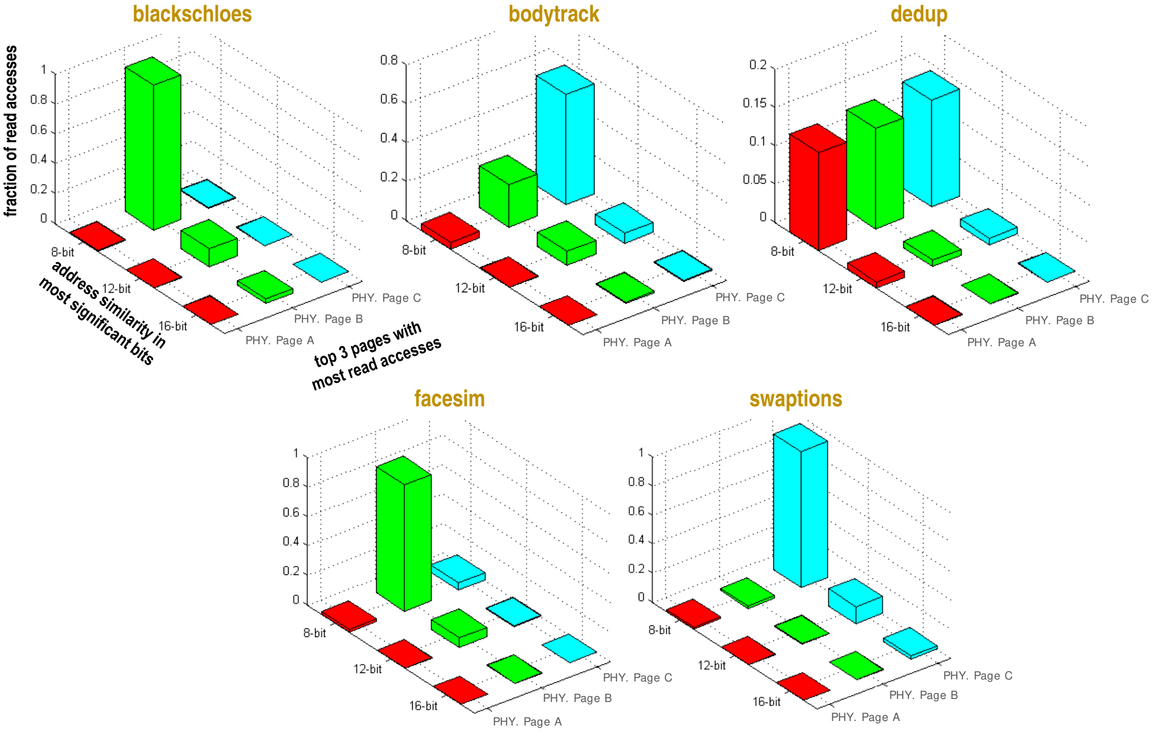


Figure 5.9: Fraction of read accesses among top three pages with the most read accesses. The x-axis shows the bin of block addresses having different number of identical MSBs within a selected page. The y-axis shows the fraction of total read accesses during the entire simulation interval. The z-axis characterizes the top three pages with the most read accesses.

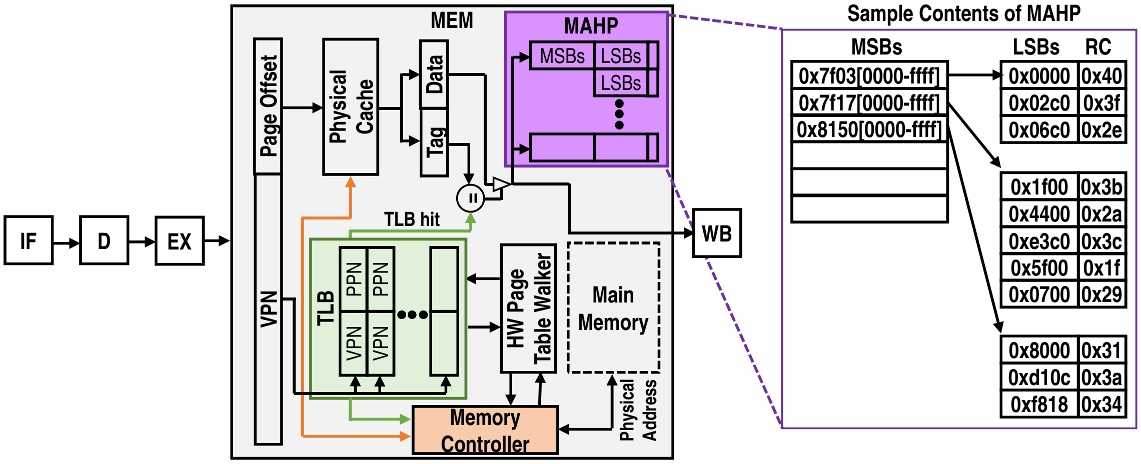


Figure 5.10: MAHP implementation in a virtual-physical cache configuration.

Algorithm 4: Multi-level Access History Profiler (MAHP)

```
Assumptions:
- RC: Read Counter, WC: Write Counter
- $NR_{th}$: read threshold level
- LRSC and HRSC are private non-LLC in core i where $i = \{1, 2, ..., number\ of\ cores\}$, L3 is shared LLC
Function MAHP-insertion() /*algorithm for inserting address of LLC block into MAHP*/
begin
    if LLC read hit from core i then
        if $RC_{block \in LLC}$ == 3 and WC == 0 then
            if $\nexists$ address of block $\in$ LLC in MAHP then
                copy address of block $\in$ LLC into MAHP
                ++$RC_{entry \in MAHP}$
            else if $\exists$ address of block $\in$ LLC in MAHP AND $RC_{entry \in MAHP} < NR_{th}$ then
                ++$RC_{entry \in MAHP}$
            else if $\exists$ address of block $\in$ LLC in MAHP AND $RC_{entry \in MAHP} == NR_{th}$ then
                copy $block \in LLC$ into HRSC$_{core\ i}$
        else
            ++$RC_{LLC\ block}$
```

5.3.4 Refresh Scheme for LRSC

The retention time of STT-RAM cell design utilized in LRSC architecture needs to be considered properly to meet the following key design issues:

1) *data stability during read operation*: The data retention time should be sufficient to retain the stability of data while cache lines are accessed during read operations, otherwise the unstable data is sensed via sense amplifier, which in turn may cause the corrupted data to be provided to the CPU. Even though the sensing resolution and reliability of sense amplifiers employed in STT-RAM cache designs influence the accuracy of the sensed data [187] [78], other characteristics such as resiliency to process variation [43], performance and power consumption [106], also play paramount roles for determining the preferred sense amplifier candidate.

2) *competitive performance delivery compared to SRAM-based L2 design*: The employment of high retention STT-RAM cells in the cache design requires a long write pulse for write operation

which results in performance degradation. This phenomenon becomes common in write-intensive applications. The retention relaxation of STT-RAM can significantly improve switching performance by reducing the data retention time, which in turn causes reduced write latency.

3) *cache block accessibility*: To stabilize the data stored in a retention-relaxed cache line, the refresh operation re-writes the cache line which has reached the end of its lifespan. However, if the interval between refreshes is considered to be short, the performance of the system may significantly decrease because the cache block for normal read and write operations is not available during its refresh cycle. In addition, increasing the refresh ratio would undermine the STT-RAM cell's endurance, which is on the order of 10^{12} [82] [129]. Thus, the refresh interval should be optimized to reduce the conflict ratio to these cache blocks while incurring insignificant refresh overhead.

To find the optimum data retention time which addresses all aforementioned challenges, a new metric called *Data Stability Interval (DSI)* is defined for each cache block. DSI is the maximum interval between a write and the final subsequent read operation before the cache block can be accessed by a write or eviction operation. For instance, Fig. 5.11 shows the entire lifetime of a cache block which has been accessed by multiple read and write memory operations over time. Since the *interval A* begins with a write operation and followed by two read operations, it has the potential to be considered as DSI. The data unstability in the *interval B* does not impact on the processing data in CPU because the data will be over-written during this period. In addition, accessing the cache line by a write operation is similar to refreshing that cache line which guarantees the stability of data up to the end of its lifespan. In *interval C*, the write operation is followed by three read accesses. The *interval C* exhibits longer period compared to *interval A*. Thus, the *interval C* is considered as DSI for this cache block. The ideal data retention time for LRSC design can be defined as follows:

$$Ideal \quad Retention \quad Time_{LRSC} = Max(DSI_0, DSI_1, ..., DSI_n) \tag{5.1}$$

where *n* is the number of cache blocks. The refresh operation can be completely eliminated for LRSC design with ideal data retention time because the data is stable over the entire concerned period of the cache blocks. Fig. 5.12 shows the ideal data retention time obtained from Eq. 5.1 for both emerging read-intensive and write-intensive workloads selected from PARSEC benchmark suite. Typically, the ideal data retention time for read-intensive workloads is protracted to satisfy the exhaustive read accesses while lengthy read-read sequences are repeatedly taken place. On the other hand, the write-intensive workloads dictate extensive write operations to the cache lines, which intrinsically leads to refreshing accessed cache lines and reduced DSI. Thus, not all cache lines need to be designed ideally for read-intensive and write-intensive workloads because of the non-uniform access behavior to the cache lines. Furthermore, the program runtime behavior is often non-deterministic when running a set of multi-threaded workloads on a multiprocessor architecture which utilizes different branch prediction techniques for performance improvement and avoids unnecessary instruction execution. To clarify this issue, we have conducted an extensive application-driven study to classify DSI distribution over time and to find the optimum data retention time by taking the energy consumption and IPC of different LRSC designs into account.

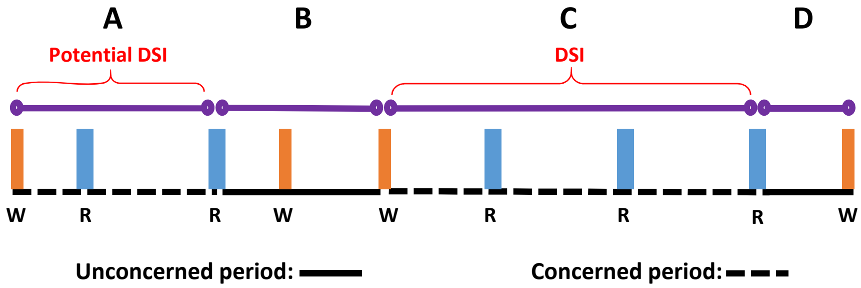


Figure 5.11: DSI equals to the sequence C which is largest interval between a write and the final subsequent read operation.

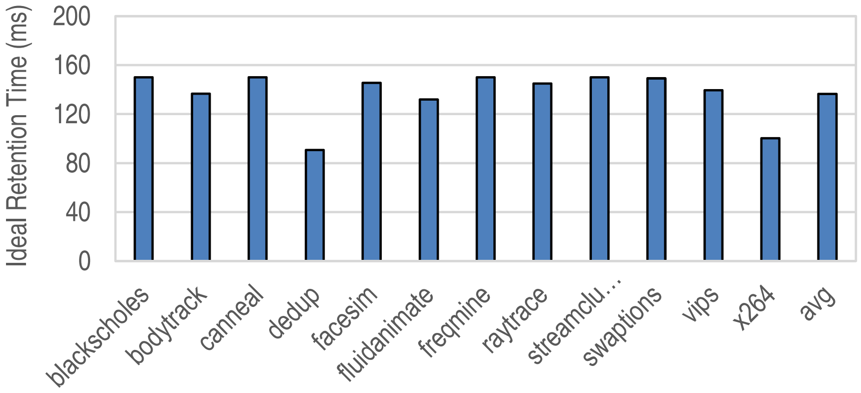


Figure 5.12: Ideal data retention time to avoid refresh scheme.

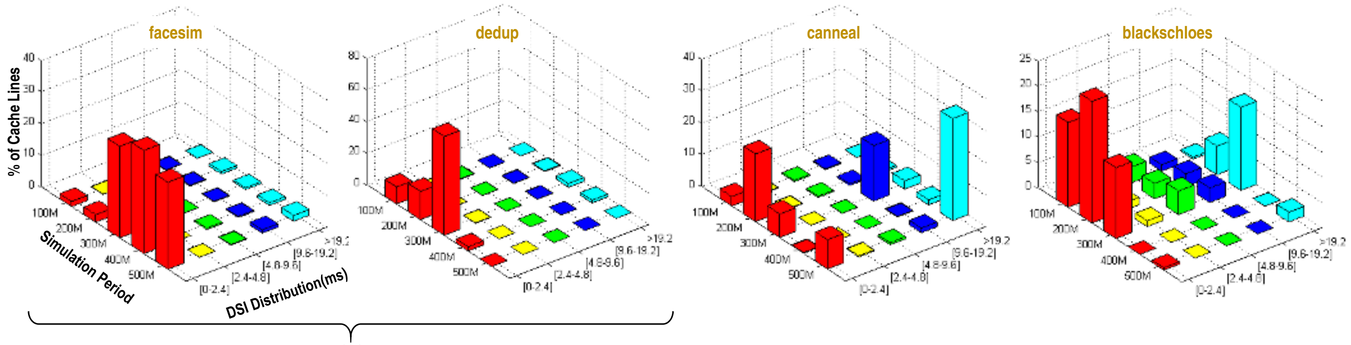
Fig. 5.13 illustrates the DSI distribution for four selected workloads. The overall simulation period is divided into five periods and the cache lines are partitioned based on their DSI dispersion. Each bar in the plot depicts what percentage of cache lines with calculated DSI are fall into which simulation period. For instance, less than 5% of all cache lines in *facesim* need to retain data longer than *19.2ms*, while this quantity goes up to 24.28% for *blackscholes* workload. Based on the non-uniformity of DSI distribution in different benchmarks, we identify three classes:

1) *Unimodal DSI Distribution*: A considerable portion of cache lines exhibit short DSI distribution for *facesim* and *dedup* benchmarks, which result in their DSI often falling into the smallest expected period that cache line must preserve data. These benchmarks require relatively narrow window retention time whereby cache lines are accessed by regular write operations.

2) *Bimodal DSI Distribution*: In this distribution, cache lines are often partitioned into two groups. The first group has short DSI while the DSI of the second group often resides in long-lasting DSI category. The *canneal* demonstrates a bimodal DSI distribution among different cache lines in which around 41% of cache lines require data retention time less than *2.4ms* while this ratio goes up to 57% for cache lines with DSI more than *9.6ms*.

3) *Symmetric DSI Distribution*: The cache lines' DSI are uniformly distributed in each DSI category over simulation periods. The benchmark *blackscholes* has approximately symmetric DSI distribution unlike the behavior of aforementioned benchmarks.

Fig. 5.14 shows the total cache lines distribution with different DSI, where the majority of cache lines (on average 70.3%) require data retention time less than *2.4ms*. On the other hand, around 18% of cache lines need long-lasting retention time more than *19.2ms* to meet data stability purpose during read operation.



(a) (b) (c)

Figure 5.13: Three classes of DSI distribution: (a) unimodal, (b) bimodal, and (c) symmetric.

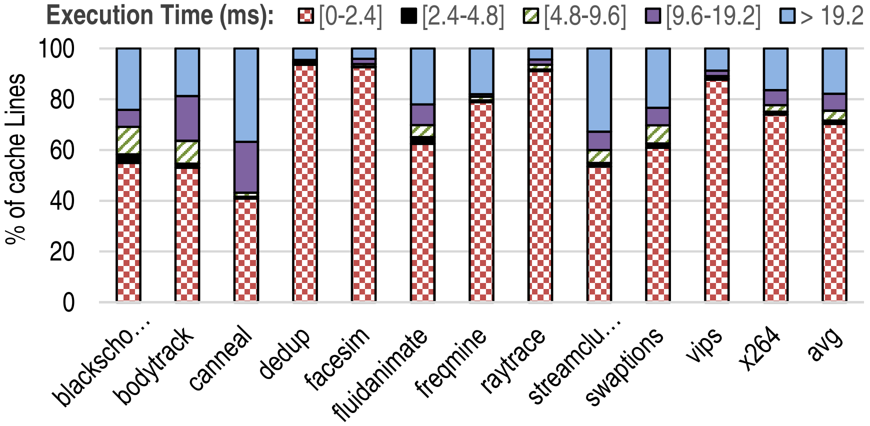


Figure 5.14: DSI distribution for all benchmarks.

To find the sufficient data retention time that accommodates provision for the above three classes while miniaturizing conflict with normal memory accesses and delivering high performance, we selected three different STT-RAM cell's retention time for comparison in terms of incurred energy consumption and offered performance as listed in Table 5.1. *Design 1* complies with the demands of both *bimodal* and *symmetric DSI distributions* to provide data retention time as high as ideal approach for cache lines with DSI more than *19.2ms*. *Design 3* offers a reduced retention time in favor of workloads with *unimodal DSI distribution* to promote write performance. *Design 2* has been considered as an intermediary to satisfy both groups of cache lines claiming either long or short DSI while also targeting cache lines having middle retention time. The reduced data′s lifespan stored in *Design 2* and *Design 3* require a refresh mechanism to prevent data loss. Therefore, the proposed refresh scheme in [161] is considered to sequentially refresh all cache blocks. We have included the energy contributions from peripheral circuits and energy consumption due to refresh mechanism in our simulation results. The detailed scheme for adjusting the retention time of the STT-RAM cell is elaborated in Section 5.3.5.

Table 5.1: STT-RAM cell retention time configurations for LRSC design.

Configuration	Design 1	Design 2	Design 3
Retention Time	140ms	10ms	1ms
Write Latency @3GHz	12 cycles	7 cycles	6 cycles

The energy consumption and IPC comparison for the above three designs are shown in Fig. 5.15 and Fig. 5.16, respectively. The relatively high retention time of *Design 1* comes at the expense of high write current and slow write speed while completely eliminating the energy dissipation and memory access conflict induced by periodic refresh operation. *Design 2* leverages the small write current realized by lowering retention time to diminish dynamic energy consumption and to improve write performance. Although it is expected to observe decent improvement in both criteria compared to previous design, the results show slight gain. The main reason is that the extra write operations associated with periodically refreshing cache lines incur additional energy dissipation while the conflict between refresh operations and normal memory accesses is trivial. The augmented number of refreshes for *Design 3* would undermine the benefits of volatile STT-RAM with *1ms* retention time whereby the conflict ratio among refresh operations and normal read/write accesses significantly increases besides advancing the write speed. Thus, since the *Design 2* offers lowest energy consumption and highest IPC among three designs, we selected it as the basis for our LRSC design. *Design 2* benefits from reduced retention time to improve energy consumption and delivers competitive write performance, while the negative impact of its refresh operations is amortized. *Design 2* reduces energy consumption by 57.09% on average compared to *Design 3* and enhances the overall IPC slightly in comparison with *Design 1*. These experimental results confirm the conducted study in [79].

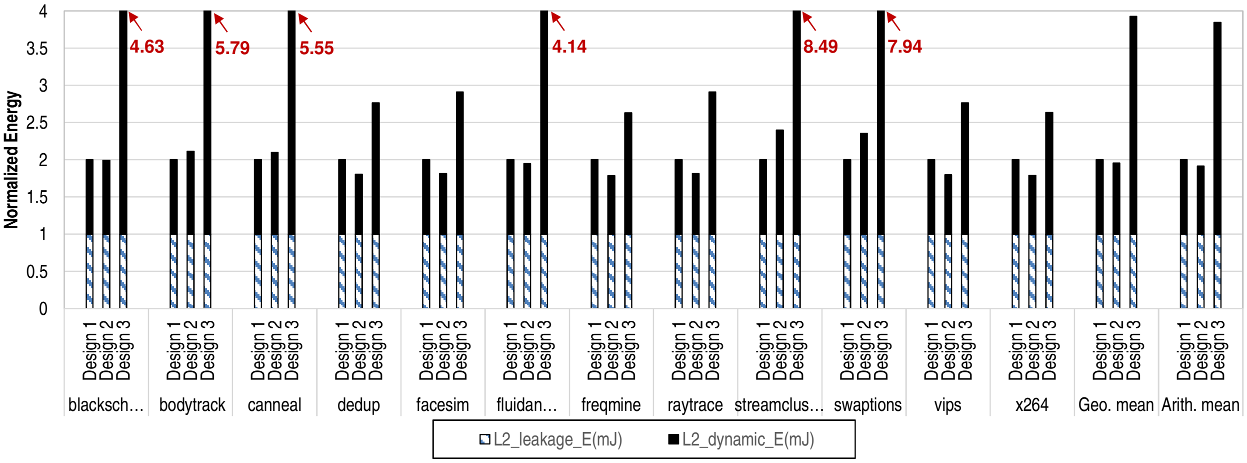


Figure 5.15: Energy breakdown comparison among three LRSC configurations normalized to Design 1.

5.3.5 Retention Relaxation in LRSC Design

STT-RAM write performance improvement relates to the optimization of retention time [79][161]. STT-RAM is usually considered as non-volatile technology, while its non-volatile characteristics

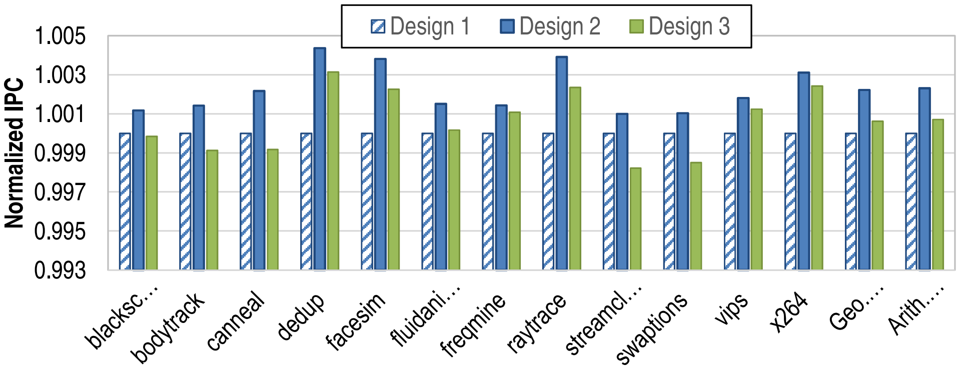


Figure 5.16: IPC comparison among three LRSC configurations normalized to Design 1.

can be relaxed to obtain better write performance. The retention time of STT-RAM, which is the period that STT-RAM can retain data until a bit-flip occurs, can be modeled as [28]:

$$t = t_1 \times e^{\Delta} \tag{5.2}$$

where t is the retention time, t_1 is is fitting constant, and Δ is thermal barrier that determines the stability of STT-RAM. The retention time of STT-RAM can be reduced exponentially by using the thermal barrier reduction whereby Δ can be characterized using [87]:

$$\Delta \approx \frac{M_s H_k V}{T} \tag{5.3}$$

where M_s is the saturation magnetization, H_k is the in-plane anisotropy field, V is the volume of free layer, and T is the absolute temperature in Kelvin.

It has been demonstrated in [87] that there are two STT-RAM switching modes: 1) *Thermal Activation (TA) region*, where I_{write} is less than critical write current (I_c) and 2) *Precessional Switching (PS) region*, where I_{write} is greater than I_c. In PS mode, the write pulse width reduction incurs the rapid increase of write current. Accordingly, some particular write pulse width can deliver optimal STT-RAM write energy. The focus of this paper is on overall write latency and energy optimization in L2 structure design. Thus, herein we adjust Δ in Eq. 5.4 to 1) calibrate the STT-RAM write speed such that it is comparable with SRAM, 2) find the optimal write current for shorter write pulse width which can still provide required write energy needed for switching in the PS region. The switching duration can be calculated as follows [173]:

$$\frac{1}{\tau_1} = \Big[\frac{2}{(C + ln(\pi^2\Delta))}\Big]\frac{\mu_B P}{em(1+P^2)}(I_{write} - I_c) \tag{5.4}$$

where τ_1 is the switching mean duration, $C = 0.577$ is Euler′s constant, μ_B is the Bohr magneton constant, P is the tunneling spin polarization of the ferromagnetic layers, e is is the magnitude of the electron charge, m is the free layer magnetic moment, and I_c is critical write current computed as below:

$$I_c = 2\alpha\frac{\gamma e}{\mu_B g}E \tag{5.5}$$

where α is Gilbert damping coefficient, γ is the Gyro-magnetic constant, g is the spin polarization efficiency factor and E is the barrier energy [87].

RRAP utilizes two distinct memory arrays to provide categories of retention times matching the required memory reference characteristics as depicted in Fig. 5.17. In our study, STT-RAM with a ten year retention time and high Δ is considered as the baseline. The volatile STT-RAM with lower Δ offers retention time of *10ms*. As shown in Fig. 5.17, two operating points HRSC (10ns, 90μA) and LRSC (2ns, 79μA) are selected. Based on the RRAP methodology, LRSC requires to be operated with low retention time while high retention time STT-RAM is needed for HRSC design. In order to reduce thermal barrier to achieve a low retention STT-RAM cell, we utilized the previous methodology proposed in previous works [79] [161] in which the planar area and thickness of STT-RAM is reduced resulting in exponentially reducing the retention time and required write current for switching. The write current, write pulse width, and the cell size associated to each point are integrated into NVSim [41] to obtain energy consumption and latency factors. The detailed device characterization of L2 cache structure are listed in Table 5.2.

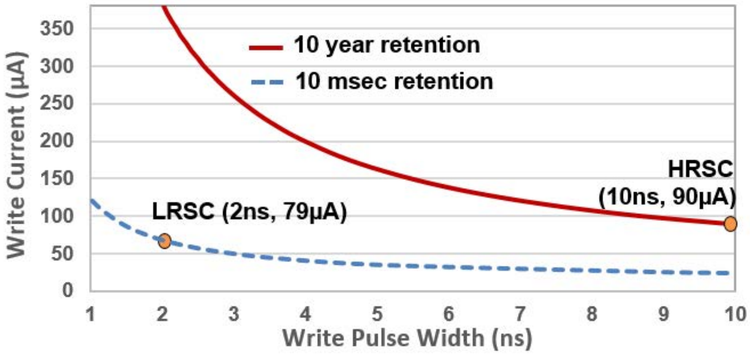


Figure 5.17: Write current vs. write pulse width for LRSC and HRSC.

5.4 Experimental Evaluation

5.4.1 Simulator Configuration

We evaluate our design using MARSSx86 [130] with PARSEC 2.1 applications. We model a Chip Multi-Processor (CMP) with eight single-threaded x86 cores. Each core consists of private L1, LHRSC L2 (LRSC and HRSC) and the LLC shared among all the cores. The detail of our model can be found in Table 5.3. Twelve applications from the PARSEC suite are selected and executed 500 million instructions starting at the Region Of Interest (ROI) after warming up the cache for 5 million instructions. The `simsmall` input sets are used for all PARSEC applications. The latency and energy usage associated with read and write operations for SRAM and STT-RAM cache accesses are provided by NVSim while DESTINY [136] is used to model eDRAM model. This is because NVSim model of eDRAM is incomplete and has not been validated. In addition, the energy contributions from peripheral circuits are also included in our simulation.

Table 5.2: Detailed characteristics of private L2 cache bank configuration (32nm, temperature=350K)

L2 Cache Technology	Area (mm^2)	RL (ns)	WL (ns)	RE (nJ)	WE (nJ)	LP (mW)
512KB SRAM	1.410	1.277	1.277	0.293	0.293	1753.444
1MB eDRAM	0.745	1.072	1.022	0.289	0.424	337.329
1MB STT-RAM	0.526	1.340	10.218	0.280	0.654	212.022
512KB LRSC	0.243	1.260	2.153	0.233	0.269	104.797
512KB HRSC	0.357	1.261	10.153	0.233	0.601	114.915

RL: Read Latency, WL: Write Latency, RE: Read Energy,

WE: Write Energy, LP: Leakage Power

Table 5.3: Memory subsystem

Processor	3GHz processor Fetch/Exec/Commit width 4
Private L1-I/D	SRAM, 32 KB, 8-way set assoc., WB cache
Private L2 Conf.	8-way set assoc., WB cache
Shared L3	eDRAM, 96 MB, 16-way set assoc., 16 bank, WB cache
Main memory	8 GB, 1 channel, 4 ranks/channel, 8 bank/rank

5.4.2 Workload Characterization for Estimating the Ratio of MSB/LSB-addresse entries

Fig. 5.18 and Fig. 5.19 show the number of required LSB-address entries per MSB-address entry for LLC blocks selected from the top three pages with the most accesses. We observed that the number of allocated LSB-address entries for a MSB-address entry may become inundated with in data-intensive workloads that have identical 12 MSBs in LLC block addresses. As an example, it is shown in Fig. 5.18 that one of the MSB-address entries of the `facesim` workload requires a high amount of LSB-address entries to determine ERRA LLC blocks. The hardware architecture support for satisfying this demand requires a complex and costly design while also incurring unacceptable energy consumption overhead to drive long search bit lines within the CAM [71, 100].

On the other hand, such pressures can be significantly reduced by moderately distributing the required LSB-address entries among MSB-address entries as illustrated in Fig. 5.19. This can be achieved by considering 16 MSBs as the MSB-address entry. Thus, the probability of allocating LSB-address entries within range of [0x00000,0xfffff] is reduced to the range of [0x0000,0xffff], which results in lowering the number of LSB-address entries per each MSB-address entry. In addition to this consideration, we proposed a dynamic LSB-address entry allocation with upper bound limit to provide sufficient space to monitor a significant portion of LLC blocks with different 16 MSBs. The upper threshold is gradually increased based on a geometric function with an exponent of 2, until it reaches the upper bound limit. In our design, the upper bound limit has been set to 512 LSB-address entries relative to each MSB-address entry. This setup has been considered after running experimental results up to three times to determine the impact of upper bound limit value on the efficiency of MAHP operation.

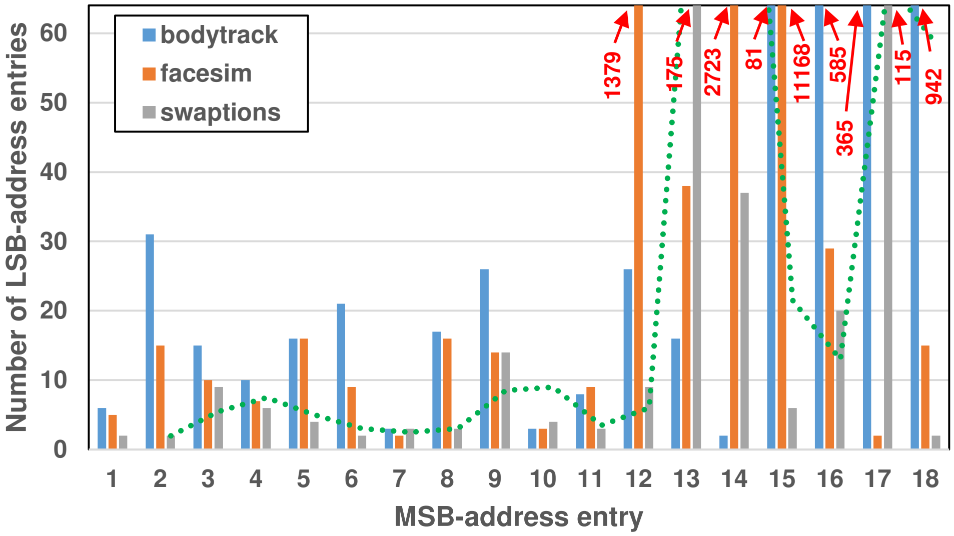


Figure 5.18: Number of required LSB-address entries per each MSB-address entry for selected LLC blocks having identical 12 MSBs.

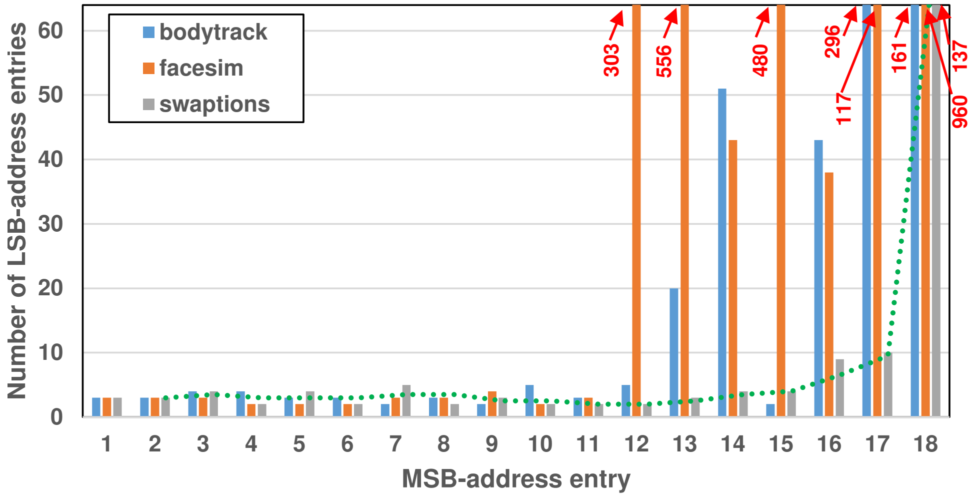


Figure 5.19: Number of required LSB-address entries per each MSB-address entry for selected LLC blocks having identical 16 MSBs.

5.4.3 Energy Usage Comparison

To comprehensively evaluate RRAP efficacy in terms of energy consumption, we compare the energy breakdown of RRAP against: `(i)` conventional 512KB SRAM-based L2, `(ii)` 1MB eDRAM-based L2, `(iii)` 1MB high retention STT-MRAM based L2, `(iv)` state-of-the-art SBAC bypass method for NVM-based L2 with *`bypassing depth=2`* [181], `(v)` state-of-the-art multi-retention hybrid cache design which is proposed in [161] and referred herein as HLR technique. The detailed characteristics of the considered technologies to be integrated into L2 design are listed in Table 5.2. Note that RRAP consists of 512KB LRSC (Design 2 in Section 5.3.4) and 512KB HRSC.

The write operation in HRSC and regular STT-MRAM requires a higher amplitude write pulse signal for retaining the data for a longer time compared to two other technologies. On the other hand, the data′s lifespan stored in LRSC is *10ms*, which makes it necessary to employ a refresh mechanism to prevent data loss. Therefore, a low-overhead refreshing scheme introduced by [161] is deployed herein which all cache blocks are refreshed sequentially. Consequently, as shown in Fig. 5.20, the consumed dynamic energy of the RRAP approach is slightly higher than the dynamic energy consumed by SRAM-based L2, but it is still significantly less than dynamic energy consumption of eDRAM which demands exhaustive refresh operation every $40us$ [28] [90]. Note that the high energy required for write operation in HRSC has negligible effect on the overall dynamic energy consumption of RRAP because the cache blocks are brought into HRSC once and are accessed by read operation for the remaining of execution time. Moreover, the high write energy and large memory cell size in regular STT-MRAM incur higher dynamic energy consumption in comparison with RRAP.

On the other hand, SBAC prevents the allocation of L2 to the blocks that occupy cache capacity without receiving sufficient cache hits. To aim this, SBAC bypasses L2 by directly providing the requested block from LLC to L1 when data with a reuse count less than *bypassing depth* is accessed. Thus, a fraction of write operations are eliminated which results in the dynamic energy consumed by SBAC becomes less than regular STT-MRAM. HLR inserts blocks into high retention region of L2 upon two incidents: 1) a read miss, 2) the lifespan of block resided in low retention region is close to the end. The large current associated with write operation in these incidents results in HLR becomes less effective rather than SBAC and RRAP. In addition, unlike RRAP, HLR adopts typical LRU replacement policy to deal with the requested blocks. This incident may cause cache thrashing which means the useful blocks become potential candidates for replacement prior to contributing sufficient read hits to L2, resulting in increased energy consumption for data movement. RRAP reduces 38.27% of the dynamic energy consumed by regular STT-MRAM which is the lowest ratio among STT-MRAM based cache designs, demonstrating that our scheme outperforms SBAC and HLR.

Fig. 5.21 compares the leakage energy consumption for aforementioned designs. Although SRAM exhibits lower dynamic energy consumption, its high leakage power has exacerbated the overall consumed energy compared to other designs. While it is widely accepted that STT-MRAM is highly persistent against leakage, its peripherals consume significant leakage energy. Thus, the regular STT-MRAM and SBAC designs offer the least leakage energy compared to two other STT-MRAM based L2 designs due to leveraging smaller-sized peripherals.

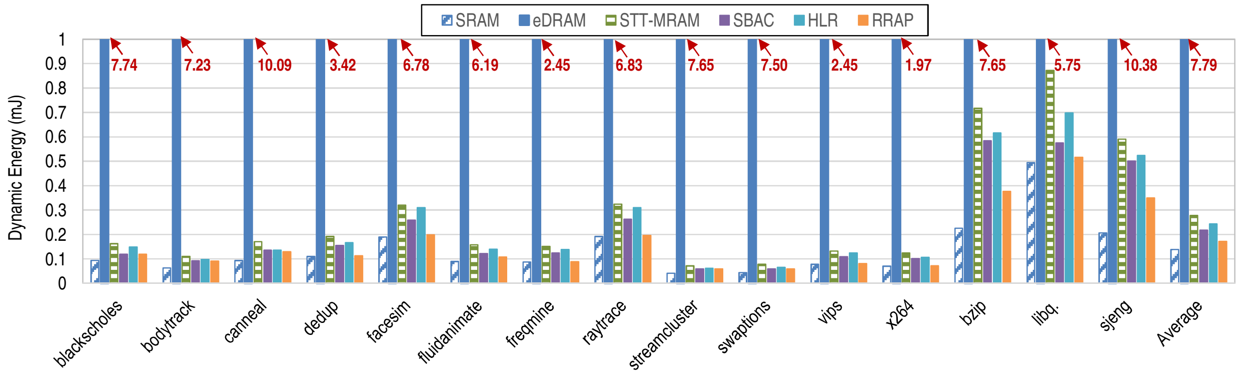


Figure 5.20: L2 dynamic energy breakdown for SRAM, eDRAM, regular STT-RAM, SBAC, HLR, and RRAP.

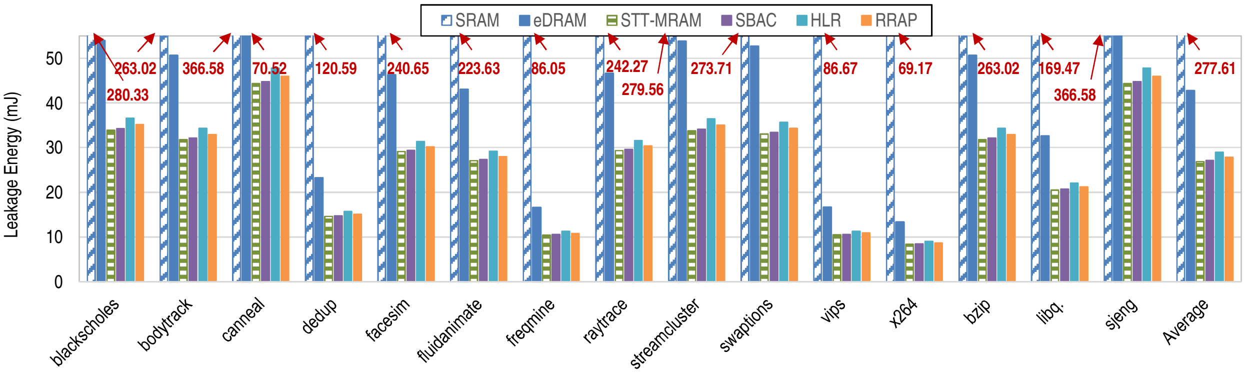


Figure 5.21: L2 leakage energy breakdown for SRAM, eDRAM, regular STT-RAM, SBAC, HLR, and RRAP.

5.4.4 Read Miss Ratio Comparison

In light of RRAP, the read miss ratio of L2 is significantly decreased as illustrated in Fig. 5.22. The main reason that RRAP can achieve the reduced miss ratio is the capability of RRAP approach to bring frequently exclusively read accessed blocks from LLC to HRSC in upper-level of cache while guaranteeing the cache blocks to reside for a large window of execution time. This accommodation significantly hides the read miss latency required to bring the frequently re-referenced data from either main memory or LLC to L2. On the other hand, SBAC incurs the greatest read miss ratio compared to other schemes because it prevents the allocation of L2 to the blocks with reuse count less than *bypassing depth*. In our experimental results, we consider the `bypassing depth=2` based on [181] which means that if a block resided in LLC is referenced twice or more, it can be inserted into L2. Hence, SBAC imposes a significant L2 miss ratio during initial placement and before that the frequently reused blocks surpass the *bypassing depth*. Furthermore, the distant interval re-reference access pattern in SBAC cache configurations, which is typical in some workloads, can exacerbate the L2 read miss ratio because enforcing the eviction of blocks before contributing sufficient read hits to L2.

RRAP reduces the read miss ratio of the regular STT-MRAM based L2 design by 51.74% on average (arithmetic mean). The experimental results show that the read-intensive workloads, such as `blackscholes`, `bodytrack` and `fluidanimate`, leverage the full potential of RRAP to further diminish the read miss ratio compared to eDRAM-based and regular STT-MRAM L2 designs.

5.4.5 Read Service Time Comparison

Herein, the elapsed time to service a read request by the processor is referred to Read Service Time (RST). To estimate the RST for the entire system, we have considered the amount of read hits and misses in all three levels of cache while also taking the trip latency for transferring a cache block from lower-level to upper-level of cache into account. SBAC offers the longest RST compared to other designs as shown in Fig. 5.23. The reason is primarily because of a high L2 read miss ratio which results in each bypassed block additionally traversing the distance between LLC and L2 compared to other designs. On the other hand, the designs which utilize the regular STT-MRAM and HLR in L2 offer nearly identical RST, but still longer RST compared to RRAP. In particular, RRAP reduces the normalized RST around 1.47% compared to regular STT-MRAM. The main reason that this improvement may seem trivial is that the service time to bring a data block from main memory to LLC contributes the most to the overall RST. This negatively impacts the efficiency of RRAP which decreases the read miss ratio in L2.

5.4.6 Performance Comparison

Besides the read miss ratio and RST comparison, we also consider IPC to compare the overall performance. Fig. 5.24 illustrates the IPC of the systems where SBAC, HLR, and RRAP exhibit greater performance compared to regular STT-MRAM designs. The main reason for performance degradation in regular STT-MRAM is its high write latency, although it can be amortized in various ways. SBAC reduces the latency associated with the write operation via avoidance of L2 allocation to non-reused blocks which results in improved performance. HLR allocates STT-MRAM arrays with low retention to write intensive blocks for faster write response which in turn results in improved performance. RRAP adjusts the HLR strategy to deal with typical access requests to L2 while also improving the temporal locality of frequently reused blocks which may exhibit distant re-reference intervals. Thus, these useful blocks are maintained long-enough in L2 to contribute sufficient L2 read hits while the required energy for preserving them is trivial. To be specific, RRAP leverages an efficient retention-relaxed STT-MRAM as LRSC to amortize the incurred high latency associated with the write operation in STT-MRAM cells. At the same time, the cache blocks are inserted into HRSC only one time while the remained memory accesses to this region are read operations.

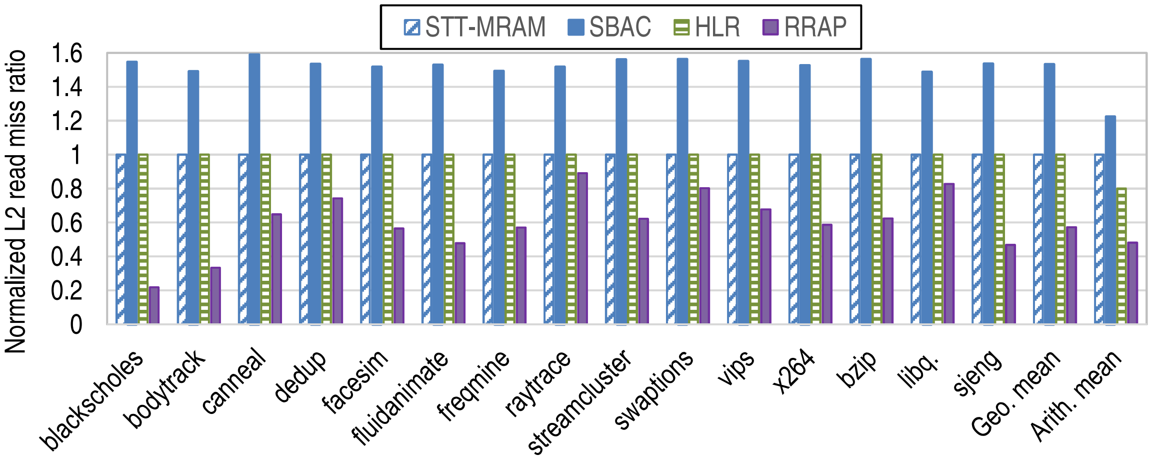


Figure 5.22: Comparison of L2 read miss ratio normalized to STT-RAM.

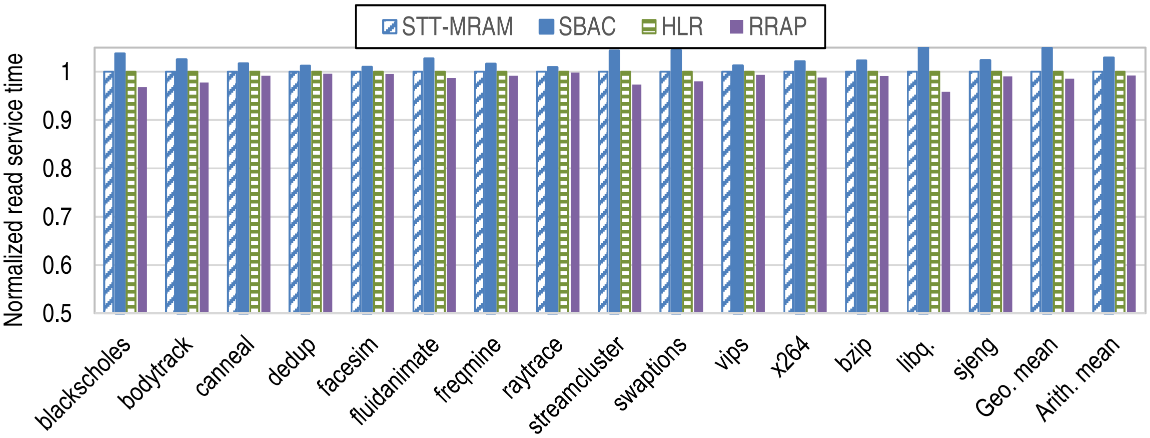


Figure 5.23: Comparison of read service time for the entire memory hierarchy normalized to STT-RAM.

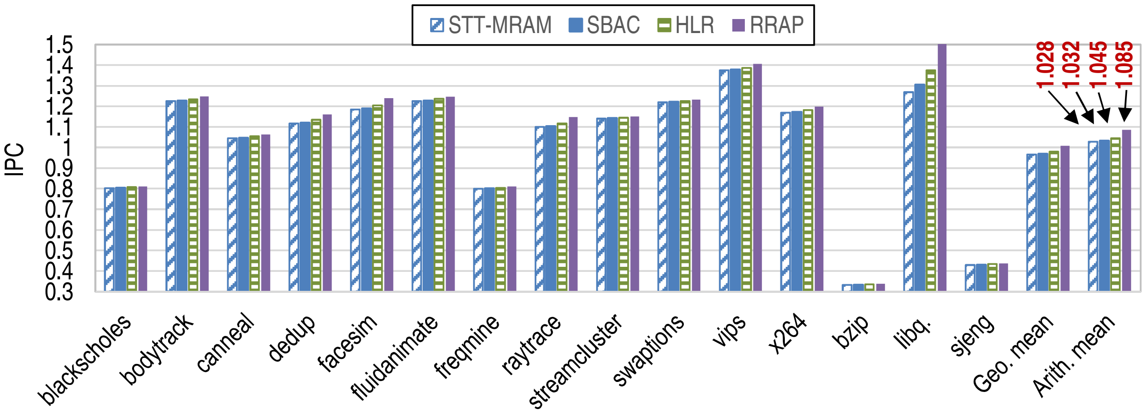


Figure 5.24: Comparison of RRAP's performance with other technologies.

5.5 Conclusions

In conclusion, RRAP retains the on-chip cache utilization close to the requesting cores for data locality maximization and lower memory access latency while taking into account the energy consumption and performance speed-up. HRSC design leverages the non-volatility of STT-RAM to store ERRA cache blocks, which in turn provides long-term residency of data without demanding extra energy to maintain the data stability. For LRSC design, we conducted an extensive exploration to find the preferred configuration in terms of energy consumption and performance. Accordingly, the write latency of LRSC is relaxed to maintain the cache utilization as high as conventional SRAM attains. Our experimental results show that RRAP can reduce the overall energy consumption and read miss ratio significantly, and read service time slightly within a comparable footprint to STT-RAM based L2 designs.

CHAPTER 6: LOGIC-IN-INTERCONNECT (LI2) COMPUTING: ENERGY-AWARE LOW-RANK MATRIX DATA IN-TRANSIT RECONSTRUCTION

To reduce further the on/off -chip interferences for accessing shared resources while eliminating unnecessary multiple round-trip to main memory for fetch and update of the data, which in turn results in the reduced orchestration workflow and data movement, we target a category of matrices that are widely observed in Big Data processing, i.e. dimension reduction, signal processing, compression, clustering, regression, and classification, called the *low rank matrix*, as a high-payoff strategy to reduce data movement across the entire system stack. In particular, the low rank matrices exhibit specic characteristics that enable computationally-efficient reconstruction of the required column using two decomposed matrices which have signicantly fewer elements than the original matrix, at the cost of a slight storage overhead. We argue that these two matrices are updated rarely in our target applications, which means that the overhead for extracting them from original matrix is correspondingly small.

The goal of this project is to realize architectural capabilities to selectively reconstruct data, instead of obliviously fetching it from lower-levels of memory, while adapting and extending highly-efficient sampling methods to significantly reduce the latency and energy cost of reconstruction. Analysis, experimental results, and simulations will be undertaken to attain the contributions below:

- LI2 strategy to significantly reduce data movement of low rank feature extraction in big data applications: While large portions of unstructured data can be represented as large low rank matrices, methods to reduce their cost of data transfer remain an underexplored research area with significant potential for achieving scalable parallelism. Thus, we propose a hardware-algorithm cross-layer solution for data on-the-fly reconstruction for energy-aware processing via ISA extensions, programmer annotations, and hardware support.

- Leverage non-volatile devices in on-chip memory units for persistently retaining the frequently accessed sparse data: To combat high leakage energy of highly-scaled CMOS, while achieving high density on-chip integration, non-volatile memory devices offer promising potential. Among such devices, Spin-Transfer Torque (STT) Magnetic Random Access Memory (MRAM) has been identified as a capable post-CMOS candidate in the 2014 ITRS Magnetism Roadmap [157], and will be utilized in the LI2 Sampling Repository to achieve vertical integration for extreme scalability.

- Adapt a scalable and robust sketching and reconstruction algorithms for data with low-dimensional structures: Building on our prior work in low-dimensional subspace recovery, our goal here is two-fold, namely, (i) draw descriptive low-dimensional sketches of the data that preserve the essential subspace information through efficient sampling designs for various data structures, (ii) devise and analyze novel techniques for subspace recovery from such sketches in the context of image and video analysis. Such techniques should be amenable to online on-the-fly reconstruction, compatible with the underlying architectures, admit parallel implementations, and reduce the data movement requirements.

6.1 Background on Memory Subsystem Design

As shown in Fig. 6.1, a modern DRAM system is comprised of a hierarchy of channels, ranks, banks, rows, and columns to exploit both the locality and the parallelism within a memory access stream. High- performance microprocessors integrate multiple DDRx channels managed by independent memory controllers. Each memory channel consists of multiple ranks that can receive

DRAM commands in parallel. Each rank comprises multiple banks organized as rows-by-columns. The banks within a rank share common data and address buses. The DDRx interface enforces a set of timing constraints between each pair of commands issued to the memory system, which makes it challenging to sustain the peak DRAM throughput. Specifically, irregular memory access patterns that require accessing multiple rows of a bank, i.e., bank conflicts, suffer from the long latency imposed by the timing constraints.

Overview of DRAM Operations: Every DRAM controller oversees managing the data movement on a single DDRx bus. It receives a request stream (reads and writes) from the cache subsystem, and generates a corresponding sequence of DRAM commands, a.k.a. command scheduling. Every request requires accessing a block of data from multiple columns of a DRAM row. Prior to a column access, a row must be brought into a *row buffer* by an *activate* command. Consecutive accesses to an active row, called row hits, enjoy the lowest access latency and can reach the peak bandwidth of the DDRx interface. A request to an inactive row requires reading the row from the bank to a row buffer, and is called a *row miss*. A row miss necessitates issuing a *precharge* command first to precharge the bit-lines of the memory array, and then issuing an activate command to activate a new row. Row misses degrade performance and increase energy consumption. DRAM accesses can be delayed due to ongoing *refresh* operations. Due to charge leakage from the DRAM cells, periodic refresh operations are required to guarantee data retention. During a refresh operation, all other accesses to the bank are blocked.

Address Mapping Strategies: A modern memory controller employs an address mapping mechanism to derive a set of DRAM coordinates–channel, rank, bank, row, and column IDs–from every physical address missed in the last level cache. DRAM address mapping has a first-order impact on system performance and energy. Existing address mapping techniques define fixed physical address bits to compute DRAM coordinates, and are largely oblivious to the underlying DRAM organization and memory reference patterns of a specific application such as low rank matrix reconstruction in the proposed LI2 scheme.

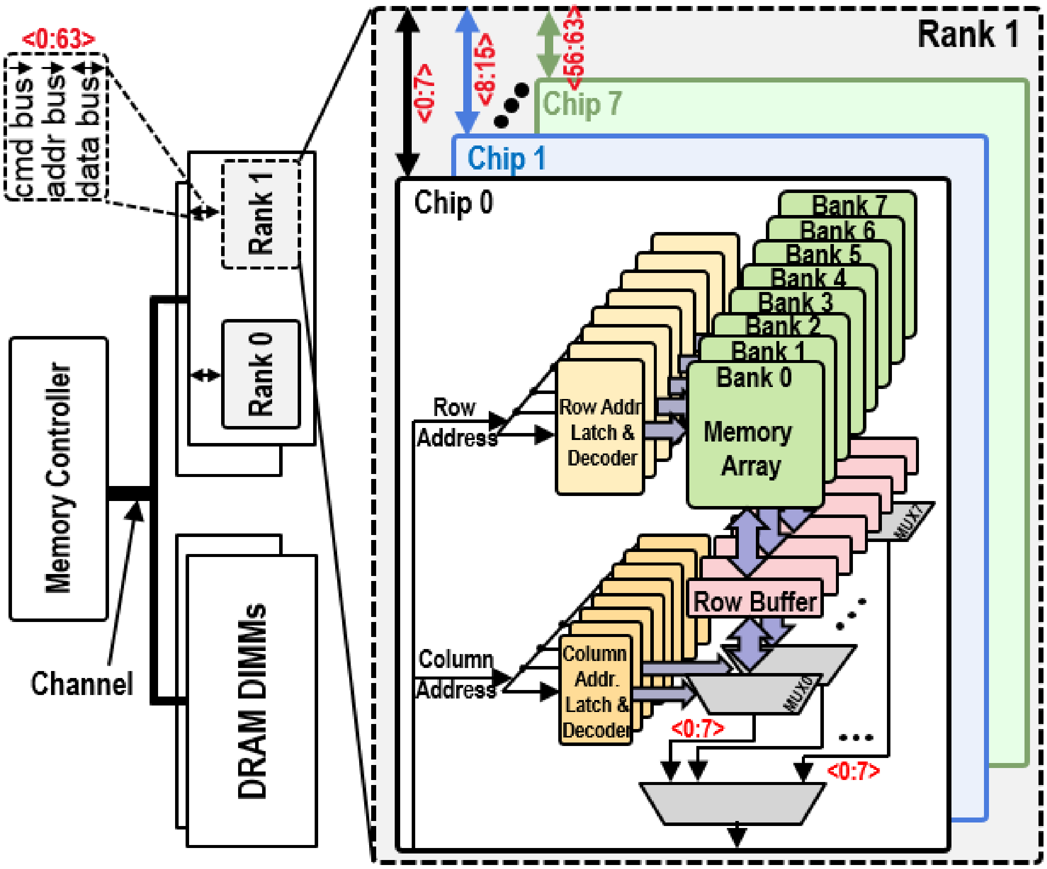


Figure 6.1: Schematic of DRAM organization.

6.2 Background on Sparse Low-rank Matrix

The sparse matrix computation has received significant attention over past years because its application in complex systems. To be specific, many use-cases in linear algebraic systems can be represented as a very large high dimensional whereby most of elements are zero. This dense matrix is referred as sparse matrix that cannot easily store and manipulate in memory. Some of the fields that sparse matrix is commonly used are as follows:

Matrix Approximation: Given a few observed entries, constructs a matrix $\widehat{M}$ that approximates M at its unobserved entries [142]. Some applications in this area are as follows:

- Linear system identification [142],
- Machine learning [54],
- Recommender System [60],
- Distance matrix completion [135], and
- Computer algebra, [119].

Non-negative Matrix Factorization (NMF): This matrix can be factorized into two matrices whereby non of three matrices have negative elements [69]. The NMF concept has application in the following areas:

- Computer vision, [69],
- Text mining, [131]
- Bioinformatics, [94],
- Blind source separation, [33],
- Unsupervised clustering, [99], and
- Natural language processing, [105].

Dictionary Learning: This method is used to represent the input data as a sparse representation [117]. In particular, it can be used in the following subjects:

- Sparse representation
- Union of subspace
- Structural sparse

The typical operations and algorithms for manipulating dense-matrix are usually inefficient when they are applied on large sparse matrices [142]. Thus, it is necessary to utilize specialized data structure and algorithms to deal with these kind of matrices. For example, some exceptional compress techniques to store sparse structure of matrix are Dictionary of keys (DOK), List of lists (LIL), Coordinate list (COO), and Compressed Sparse Row (CSR). Even though the compress techniques may significantly reduce the size of matrix that must be stored, the spacious nature of these matrices causes that the main memory to be used as the steady storage component.

On the other hand, the employment of memory hierarchy in the modern computer architecture aims to improve program's data locality which in turn directly impacts the performance of the system. To be specific, when a piece of data is still in the cache, there is no need to wait few hundred cycles till the requested data to be fetched from main memory. If the memory accesses to the program's data structure follows a certain rules, it is possible to improve temporal and spatial locality. For example, by modifying the cache replacement policy for maintaining a portion of program's data structure that frequently is accessed, the performance can be significantly improved. Many applications show temporal or spatial locality behavior which means that they are able to benefit from previously proposed methods for improving cache utilization.

However, the large sparse matrix is often indirectly accessed during program execution. This means that it is hard to predict which element of the matrix will be referenced in the next cycle. This issue makes the existing prediction and prefetch techniques inefficient. Even though there are some algorithms that aim to reduce matrix computation by modifying the data structure of matrix prior to execution, there is still a gap between software techniques and hardware capability to accelerate service to matrix computation. In particular, the challenges of indirect accesses to a large sparse matrix located in main memory are:

- the main memory bandwidth limitation to handle the memory requests,
- bus contention due to high traffic of requests and responds on the bus,
- the large latency to access the data located in off-chip memory, and
- the high power dissipation associated with off-chip main memory access while exchanging data (off-chip access consumes 250x more energy than on-chip cache [18]).

This dissertation targets a category of sparse matrices called low-rank matrix for reducing data movement in the entire computer system. In particular, the low-rank matrices exhibit specific characteristics that enable us to reconstruct the required column using two decomposed matrices which have significantly fewer elements than the original matrix. The context diagram of the proposed approach which is called *Logic-In-Interconnect (LI2)* is illustrated in Fig. 6.2.

As shown in this figure, we propose to use NVM technology to store two decomposed matrices. We argue that these two matrices are updated rarely in our target applications which means that the overhead for extracting them from original matrix is trivial. To be specific, we will employ STT-MRAM based Scratchpad component coupled with LLC to load these sub-matrices. An example of matrix decomposition process is shown in Fig. 6.3.

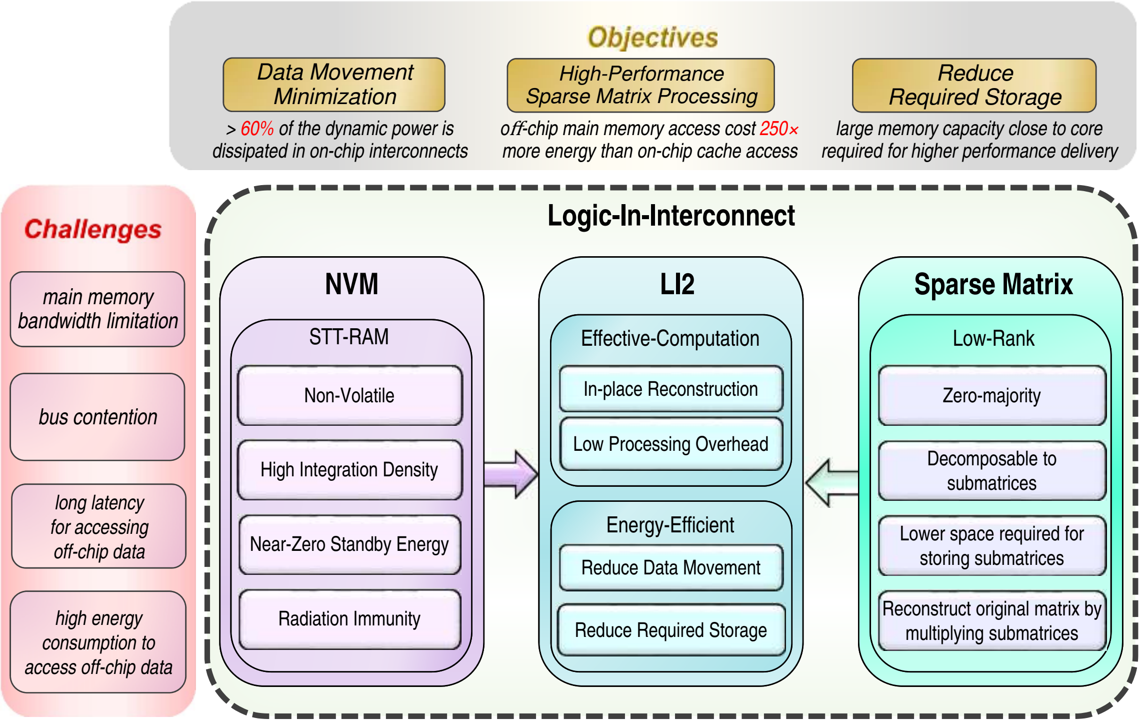

Objectives
Data Movement Minimization
> 60% of the dynamic power is dissipated in on-chip interconnects
High-Performance Sparse Matrix Processing
off-chip main memory access cost 250× more energy than on-chip cache access
Reduce Required Storage
large memory capacity close to core required for higher performance delivery
Challenges
main memory bandwidth limitation
bus contention
long latency for accessing off-chip data
high energy consumption to access off-chip data
Logic-In-Interconnect
NVM
STT-RAM
Non-Volatile
High Integration Density
Near-Zero Standby Energy
Radiation Immunity
LI2
Effective-Computation
In-place Reconstruction
Low Processing Overhead
Energy-Efficient
Reduce Data Movement
Reduce Required Storage
Sparse Matrix
Low-Rank
Zero-majority
Decomposable to submatrices
Lower space required for storing submatrices
Reconstruct original matrix by multiplying submatrices

Figure 6.2: Context diagram of challenges that Logic-In-Interconnect can overcome.

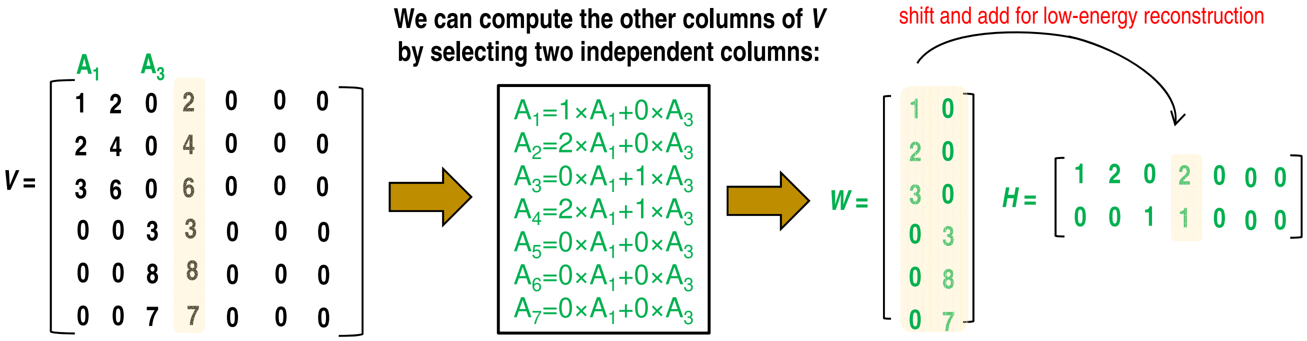

A_1
A_3
$V=$
1 2 0 2 0 0 0
2 4 0 4 0 0 0
3 6 0 6 0 0 0
0 0 3 3 0 0 0
0 0 8 8 0 0 0
0 0 7 7 0 0 0
We can compute the other columns of V by selecting two independent columns:
$A_1=1\times A_1+0\times A_3$
$A_2=2\times A_1+0\times A_3$
$A_3=0\times A_1+1\times A_3$
$A_4=2\times A_1+1\times A_3$
$A_5=0\times A_1+0\times A_3$
$A_6=0\times A_1+0\times A_3$
$A_7=0\times A_1+0\times A_3$
$W=$
1 0
2 0
3 0
0 3
0 8
0 7
shift and add for low-energy reconstruction
$H=$
1 2 0 2 0 0 0
0 0 1 1 0 0 0

Figure 6.3: An example of matrix factorization.

The theorem of matrix decomposition will be presented in detail in the next section, but a brief notation is:

- the concept of 'sparse coding' refers to a representational scheme where only a few units (out of a large population) are effectively used to represent typical data vectors [69],

- the rank of a matrix is defined as the maximum number of linearly independent column vectors in the matrix [69],
- let V be a m-by-n matrix of rank r. V can be decomposed into a m-by-r matrix W and r-by-n matrix H such that

$$V = W \times H \tag{6.1}$$

where $r << min(m, n)$. According to aforementioned theorem, $m \times n$ elements are required for storing V while the required memory elements for storing $W \times H$ is $r \times (m+n)$. Thus, the storage for saving $r \times (m + n)$ is significantly less than storage for $m \times n$.

6.3 Approaches and Algorithms for Low Rank Matrix Recovery

Addressing the ’curse of dimensionality’ with low-dimensional structures: In order to leverage the proposed hardware methods, algorithms addressing fundamental challenges facing contemporary data science must be employed. The so-called curse of dimensionality, which pertains to mining voluminous and high-dimensional data, sometimes data comprising billions of dimensions and a number of samples of the same order, poses significant challenges in terms of both computational complexity and memory requirements. This has motivated noteworthy research efforts on dimensionality reduction [83] and low-dimensional embedding [80]. In this project, we seek to leverage low-dimensional structures intrinsic to much of the real-world datasets to enable memory-efficient data management solutions and fast inference algorithms. In particular, we will focus on problems where the high-dimensional data matrices are low rank.

Low rank matrices: To clarify, let $\mathbf{L} \in \mathbb{R}^{N_1 \times N_2}$ denote a data matrix. The column and row spaces of $\mathbf{L}$ are defined as the span of its columns and rows, respectively, with equal dimension, r, called the rank of $\mathbf{L}$ [36]. The rank of $\mathbf{L}$, also defined as the number of its non-zero singular values, can be obtained through simple Singular Value Decomposition (SVD) of $\mathbf{L}$ [36]. The matrix $\mathbf{L}$ is called a low rank matrix if $r \ll \min\{N_1, N_2\}$, corresponding to a small number of degrees of freedom. The columns/rows of a low rank matrix lie in a low-dimension linear subspace of the otherwise high-dimensional ambient space. This particular feature enables us to devise scalable solutions and remarkably fast algorithms that leverage this low-dimensional structure for saving, processing, and recovering low rank matrices. Next, we briefly outline some of the proposed approaches and algorithms to leverage the low-rankness of the data, along with a description of our data sketching and subspace learning approach, which will be exploited in this project in the context of data management and recovery techniques.

6.3.1 Saving Low Rank Matrices

The SVD of matrix $\mathbf{L}$ with rank r ca be expressed as

$$\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \,, \tag{6.2}$$

where $\mathbf{U} \in \mathbb{R}^{N_1 \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, and $\mathbf{V} \in \mathbb{R}^{N_2 \times r}$ are the matrix of left singular vectors, the diagonal matrix of singular values, and the matrix of right singular vectors, respectively. Hence, the whole low rank matrix can be reconstructed through a couple of low complexity multiplication operations of low dimensional matrices $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{\Sigma}$. Accordingly, the memory requirement to store the low rank matrix is $\mathcal{O}(\max\{N_1, N_2\}\, r)$ which can be significantly smaller than $\mathcal{O}(N_1 N_2)$ in big data regime. The complexity of the singular value decomposition is roughly $\mathcal{O}(N_1 N_2 r)$. An efficient reconstruction formula, where the low rank matrix is obtained as the multiplication of two low dimensional matrices, is not limited to the singular vector representation (6.2). The low rank matrix can be recovered using a small subset of its columns/rows. To clarify, let $\mathbf{R}_1 \in \mathbb{R}^{N_1 \times n_2}$ denote a subset of the columns of $\mathbf{L}$ and $\mathbf{R}_2 \in \mathbb{R}^{n_1 \times N_2}$ a subset of the rows of $\mathbf{L}$. Define $\mathbf{C} \in \mathbb{R}^{n_1 \times n_2}$ as the sub-matrix of $\mathbf{L}$ constructed from the intersection of $\mathbf{R}_1$ and $\mathbf{R}_2$. If the columns of $\mathbf{R}_1$ span the column space of $\mathbf{L}$ (i.e., the rank of $\mathbf{R}_1$ is equal to r) and the rows of $\mathbf{R}_2$ span the row space of $\mathbf{L}$ (i.e., the rank of $\mathbf{R}_2$ is equal to r), then the low rank matrix can be obtained [101] as follows:

$$\mathbf{L} = \mathbf{R}_1 \mathbf{G} \mathbf{R}_2 \,, \tag{6.3}$$

where $\mathbf{G} \in \mathbb{R}^{n_2 \times n_1}$ is a matrix obtained directly from $\mathbf{C}$ through low complexity operations. The values of n_2 and n_1 can be as small as r. However, there is a caveat; finding the informative subset of the columns of $\mathbf{L}$ (which span its column space) may be itself computationally expensive [68]. Nonetheless, it turns out we can construct matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ using random column/row sampling. The sufficient number of randomly sampled columns/rows that span the column/row-space of $\mathbf{L}$ scales linearly with r [139]. We will leverage this approach in this project to effectively save and reconstruct large data matrices to reduce off-chip memory traffic.

6.3.2 Recovering Low Rank Matrices

In this subsection, we identify how the low-rankness of the data (or a component of the data) can enable fast recovery even in presence of severe corruptions and missing values in the context of two well-known problems, namely, matrix completion and low-rank-plus-sparse-decomposition. These techniques will be integrated within this project at the level of hardware for efficient data handling and inference.

Low Rank Matrix Completion: Suppose we only observe a random set of the entries of a low rank matrix $\mathbf{L}$. The question is whether one can somehow complete the matrix, i.e., recover the unobserved entries from the observed ones given the side information that $\mathbf{L}$ is low rank. This question is relevant to many practical applications including recommender systems where users submit ratings for few select items and the vendor provides recommendations by inferring their preferences for unrated items [3]. Another application is localization through triangulation from incomplete data [156]. In [26], the authors answered this question in the affirmative by proposing a convex program for matrix completion that can provably recover the low rank matrix correctly from a fairly small number of randomly sampled elements with high probability provided that the column and row spaces of $\mathbf{L}$ are incoherent with the standard basis. This condition essentially requires that the column and row spaces are not aligned with any of the main directions of the canonical coordinate system, which amounts to $\mathbf{L}$ not being simultaneously low rank and sparse. The sufficient conditions presented in [26] require roughly $\mathcal{O}(r \max(N_1, N_2)^{1.2})$ randomly-sampled elements to guarantee correct matrix completion, however, the numerical experiments show that even a smaller number of measurements can yield exact recovery. The lesson learned is that the data is still recoverable despite the fact that a remarkable portion of the data could be missing provided that the data is low rank and that the low rank matrix is not sparse.

Low Rank Matrix Recovery in Presence of Sparse Corruptions: In many applications, the given data $\mathbf{D} \in \mathbb{R}^{N_1 \times N_2}$ can be represented as:

$$\mathbf{D} = \mathbf{L} + \mathbf{S} \tag{6.4}$$

where $\mathbf{L}$ is an unknown low rank matrix (with unknown rank) and $\mathbf{S}$ is an unknown sparse matrix with completely unknown support [27, 25, 140]. Applications of this model include video surveillance and face recognition, just to name a few [25]. For example, in video surveillance we may be interested in detecting activity (captured by the sparse component $\mathbf{S}$) against a slowly changing background (the low rank component $\mathbf{L}$). Interestingly, although we have no information about neither the low rank nor the sparse components beyond this structural information, the data can be decomposed correctly into its correct low rank and sparse components provided that the rank of $\mathbf{L}$ is sufficiently small. In [27, 25], the authors devised a convex algorithm which was shown to yield exact decomposition if the rank of $\mathbf{L}$ is sufficiently small, the column space and row space of $\mathbf{L}$ are sufficiently incoherent with the standard basis, and the support of $\mathbf{S}$ is sufficiently diffused.

Data Sketching and Subspace Learning Approach: In our recent works [139, 140],further results indicate that a low rank matrix can be correctly recovered in the presence of sparse corruptions even if only $\mathcal{O}(r\max(N_1, N_2))$ columns/rows of $\mathbf{D}$ are available. The key idea underlying our approach to achieve such gains is subspace learning in small data sketches. In particular, small data sketches can provably preserve the structural information in the data given its limited number of degrees of freedom (low-rankedness). This fact allows us to address the inference problems of interest by solving low complexity optimization problems – essentially the complexity only scales with the number of degrees of freedom rather than the underlying data dimensions. In addition, inference can be carried out using a small subset of the data. This has huge implications on memory usage and data exchange at the hardware level since we no longer need to use or exchange the full-scale data, which entails substantial speedups. As a concrete example, we are able to solve the low rank matrix recovery in presence of sparse corruptions in few milliseconds versus half an hour for the direct approach that uses the full-scale data on a standard 3.4GHz CPU in the context of activity detection and background subtraction from real video frames, which corresponds to more than a 1000-fold speedup in data processing. The aforementioned examples underscore the effectiveness of data sketching in recovering low rank matrices using scalable and provable algorithms. We will explore and leverage these techniques to reduce data traffic, in the context of the applications described in this project.

6.4 LI2 Scheme to Realize Data Reconstruction

Fig. 6.4 illustrates the clear-box view of a feasible approach for integrating LI2 into the memory architecture. LI2 tightly communicates with Memory Controller and Hardware Page Table Walker for extracting the address of programmer annotated data samples while fetching and placing data samples into the Sampling Repository located in Request Translation component of LI2 as illustrated in Fig. 6.5. Since the sampled elements of the low rank matrix are updated infrequently, it is proposed to embed STT-MRAM as an appropriate emerging technology to realize a near-zero leakage power, while imparting soft error resilience and supporting area-saving vertical integration for implementing Sampling Repository is targeted for researching the application of non-volatile memory. The Last Level Cache (LLC) miss on data from a low rank multi-dimensional array triggers LI2 to reconstruct the requested data instead of fetching it. LI2 scheme alleviates the memory pressure by reducing the off-chip memory requests while processing a low rank matrix data array. Thus, the freed memory bandwidth can be allocated to transfer unconstructable workloads between main memory and the processing core, which results in increasing the potential scalability of large-scale applications. To equip the hardware with LI2 computation support, we adopt the conventional address translation semantic that also supports updating tag store of Sampling Repository in LI2 during loading programmer annotated data samples. Upon receiving a request from the Memory Controller to reconstruct data, LI2 supplies the Multiplier Stream with the corresponding data samples. The results of each multiplication are summed and returned directly to the processing core.

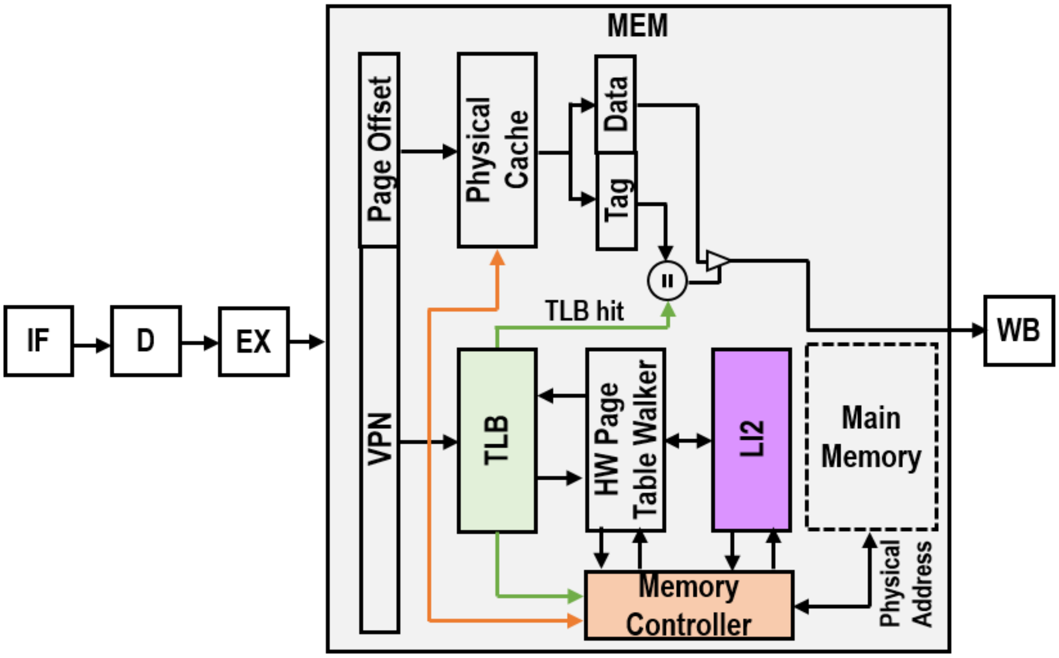
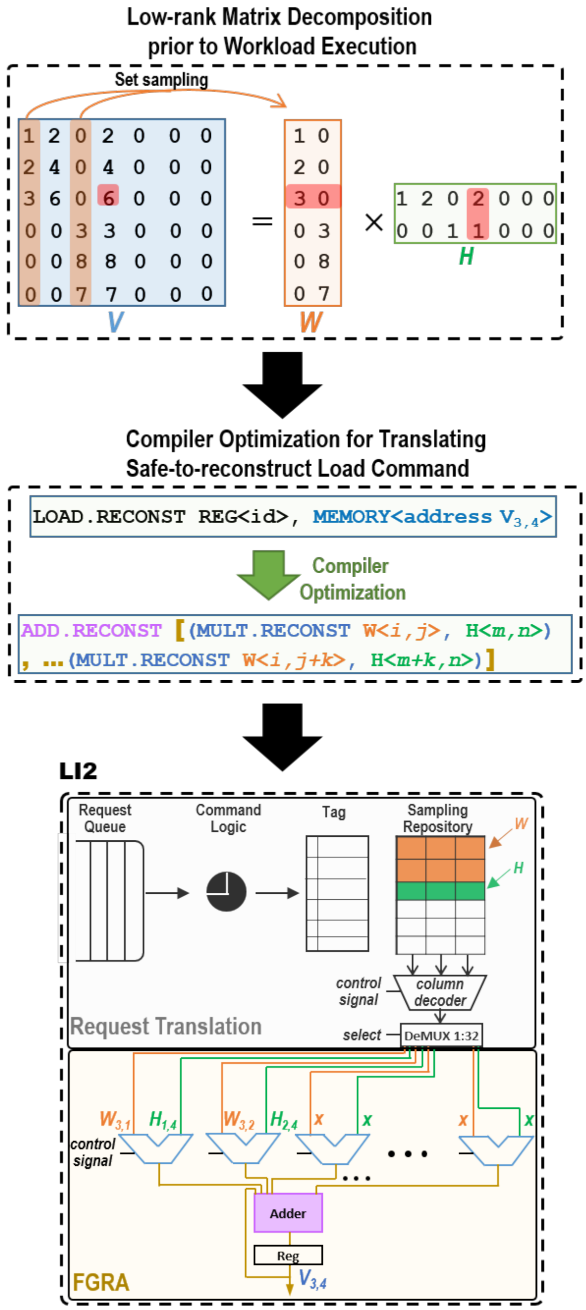


Figure 6.4: LI2 implementation feasibility in a virtual-physical cache configuration. LI2 is responsible for reconstructing data via the memory controller.



$$\underbrace{\begin{bmatrix} 1 & 2 & 0 & 2 & 0 & 0 & 0 \\ 2 & 4 & 0 & 4 & 0 & 0 & 0 \\ 3 & 6 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 3 & 3 & 0 & 0 & 0 \\ 0 & 0 & 8 & 8 & 0 & 0 & 0 \\ 0 & 0 & 7 & 7 & 0 & 0 & 0 \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \\ 0 & 3 \\ 0 & 8 \\ 0 & 7 \end{bmatrix}}_{W} \times \underbrace{\begin{bmatrix} 1 & 2 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}}_{H}$$

```
LOAD.RECONST REG<id>, MEMORY<address V3,4>
```

```
ADD.RECONST [(MULT.RECONST W<i,j>, H<m,n>)
, ...(MULT.RECONST W<i,j+k>, H<m+k,n>)]
```

Figure 6.5: Illustrative example of LI2 processing.

6.4.1 ISA Extension for LI2 Realization

LI2 relies on Instruction Set Architecture (ISA) extension and programmer annotations to enable the compiler to determine the safe-to-reconstruct data arrays, set sampling data, and standard accesses. The same approach has been used in the previous works for approximate computing. Thus, we will research tradeoffs of LI2 memory bus contention and read/write request latency compared to an unenhanced baseline. As illustrated in Table 6.1, an initial ISA extension utilizing two bits in the opcode can be set to command the microarchitecture load/store and computation operation. In particular, executing `LOAD.SAMPLE REG<id>, MEMORY<address>` fetches the data from `MEMORY <address>` to `REG<id>` while also initiating the Sampling Repository. By executing (`LOAD.RECONST REG<id>, MEMORY <address>`) if the data is not found in cache hierarchy, the memory controller sends this request to LI2 for supplying the reconstructed data to the processing core. All `ALU.RECONST` operations offload the arithmetic operations onto FGRA resided in LI2, as described later.

Table 6.1: Illustrative example of ISA extension to support LI2 realization.

Additional bits in opcode	**Instruction type**	**Description**
00	**ALU**(`ADD, SUB, SLL, SRL, AND, OR, XOR,NOT`), **Control Flow**(`JMP, BNEQ, BTQE`), **Data Memory** (`LOAD, STORE`)	**standard access**
01	`LOAD.SAMPLE, STORE.SAMPLE`	**set sampling data**
10	`LOAD.RECONST, STORE.RECONST`	**safe-to-reconst. load/store data**
11	`ALU.RECONST(MULT.RECONST, ADD.RECONST)`	**safe-to-reconst. arithmetic data**

6.4.2 LI2 Implementation Steps

Programmer annotation aids the compiler to safely optimize the instructions to be executed on corresponding computation components. To target this, prescreening of the data layout will be assessed to determine data samples and to extract two sub-matrices for reconstructing data without accessing off-chip memory. After extracting the required information, the extended compiler will convert memory loads from the safe-to-reconstruct matrix in main memory to the corresponding `LOAD.RECONST` command. In addition, each `LOAD. RECONST` will be further converted to `ADD.RECONST` command that determines which elements from two sub-matrices must be multiplied, and finally summed up with others. When the processor executes one of safe-to-reconstruct arithmetic operations, the data processing is offloaded to the LI2 unit. As illustrated in Fig. 6.5, the $V_{3,4}$ data entry can be reconstructed by multiplying the third row of W with the fourth column of H. Thus, the compiler converts `LOAD.RECONST REG<id>, MEMORY<address of` $V_{3,4}$`>` to multiplication and accumulation operations on associated data samples. The requests from the processing core are enqueued in a Request Queue buffer before processing by command logic. The requests are dequeued with FCFS ordering. The Command Logic uses the compile-time information to fetch the associated elements from the Sampling Repository and sends the stream of corresponding data to the Multiplier Stream. The column decoder and demux will lead each entry to the corresponding input in the FGRA. Upon LI2 process completion, the reconstructed data will be returned to the processing core. To maximize the speedup of supplying inputs to FGRA, we will research designs for the Sampling Repository in such ways so that entries of W and V reside adjacent to each other for increasing special locality. Thus, by an access to the associated cache block, 16 entries of either W or V will be read,

6.4.3 Fine Grained Reconfigurable Array (FGRA)

It is proposed to provide 16 power-gated multipliers in the FGRA, which can be reconfigured based on the demand for multiplication operations. If more than 16 multiplication operations are required, then the results for the first 16 multiplications will be stored in a register file, and it will be summed up with the remainder of multiplications in the next clock cycle. Eventually, the data will be reconstructed without attenuation. This data will be supplied to the core without storing it in a higher level of cache. However, since the application may need to reconstruct it again in the future, a replica copy of reconstructed data will be retained in a 16-entry buffer which retains the history of the last 16 reconstructed data elements. The primary reason for skipping the caching process of reconstructed data is that (1) the probability of re-accessing to this data is rare in the future, (2) only 4B of 64B cache block will be available after each LI2 operation. Accordingly, we will research efficient designs for retention to optimize the hit rate within LI2 strategies, in addition to the proposed 16-entry buffer for maintaining recently-reconstructed elements.

6.4.4 LI2 Cache Coherence Protocol

The Sampling Repository operates as another level of the cache hierarchy. The main reason for implementing an additional level of cache in FGRA arrangement is to guarantee that the samples of low rank matrix are persistently-accessible during data reconstruction. The baseline Sampling Repository will be designed as a direct-mapped cache with the same cache block size as LLC design. To ensure that the sample dataset is consistent across the entire cache hierarchy, an exclusive policy can be enforced.

6.5 Preliminary Results

Based on our previous study [155], our design for dual-rail pipelined *Modified Baugh-Wooley* multiplication algorithm performs a 32-bit 32-bit multiplication in average cycle time of 12.7ns for 0.25m technology node. The optimal multiplier design needs to be investigated to further reduce the multiplication time during reconstruction operation, considering the high impact of CMOS technology scaling on reducing the gate delay and power consumption. This includes the investigation and redesign of high-performance multiplication algorithms such as *modified Baugh-Wooley*, *Wallace Trees*, *modified Booth*, and *Array Multiplier with Carry Propagate Adders (CPAs)*.

In LI2, the number of multiplication operations has linear relation with the rank of low rank matrix as illustrated in Fig. 6.6. For example, face images having different illumination are known to lie in a low dimensional subspace, approximately 8 dimensional, which means that a 32-bit data can be reconstructed by summing the results of eight 32-bit 32-bit multiplications. Fig. 6.7 shows an estimation of energy consumption for data reconstruction in FGRA unit based on the type of required arithmetic operations, indicating that LI2 can provide 4.0x to 6.5x decrease in energy usage over recent 3D-stacking DRAM solutions consuming around 5.1 pJ per bit access while also requiring 0.75 pW power per bit during standby mode [61]. In addition, LI2 eliminates the high energy cost for periodic refresh operation to maintain the original low rank matrix in DRAM by guaranteeing to recover the original data through subspace recovery algorithms.

Following [13], it is expected to achieve an average cycle time of 5 ns and 6.25 ns or even less to perform a 32-bit 32-bit Floating-Point (FP) multiplication and 64-bit FP addition, respectively, which means a cache block in a low rank matrix data layout having rank of less than 16 can be reconstructed approximately in around 180 ns = (16 [5 ns + 6.25 ns]) for FGRA unit as illustrated in Fig. 6.8, which is 10% less than the average cycle time for transferring a cache block from off-chip memory subsystem to the processing core [40, 127].

We anticipate that LI2 can further reduce data transfer cost in applications with irregular accesses in which the row buffer of memory's bank must be reloaded with new row. Thus, LI2 can significantly alleviate bus contention and demand on the main memory subsystem.

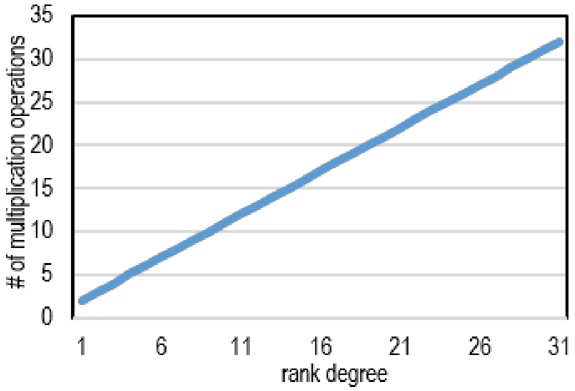


Figure 6.6: Linear relation of rank degree and multiplication operations.

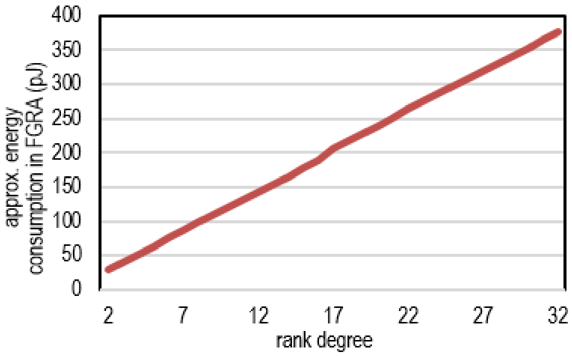


Figure 6.7: Energy consumption estimation for data reconstruction where energy per FP MUL and per FP ADD considered to be 11.3 pJ and 7.1 pJ, respectively.

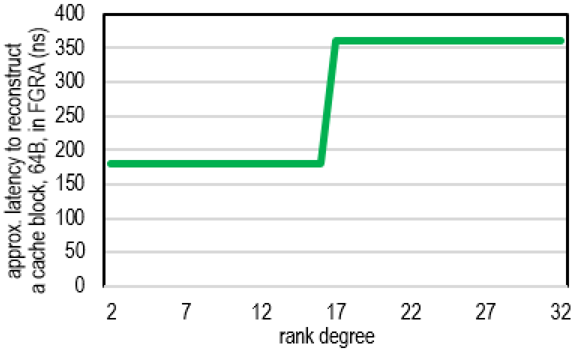


Figure 6.8: Latency estimation for data reconstruction where latency per FP MUL and per FP ADD considered to be 5 ns and 6.25 ns, respectively.

6.6 Conclusions

The goal of this project is to realize a hardware/algorithm cross-layer solution to reconstruct data, instead of obliviously fetching it from lower-levels of memory in the workloads dealing with low-rank matrices, while adapting and extending highly-efficient sampling methods to significantly reduce the latency and energy cost of reconstruction.

CHAPTER 7: CONCLUSION

The increasing bandwidth demand for memory-intensive workloads has motivated both industrial and academia to investigate hardware/software/algorithm cross-layer solutions to reduce data transfer between on-chip and off-chip memory. One way to accommodate larger portion of workload in on-chip cache is to employ larger last level cache. In particular, eDRAM and STT-MRAM memory technologies have received significant attention for this purpose due to offering area and energy beneficiary rather than SRAM. However, there are some reliability concerns associated with each of these technologies.

eDRAM suffers from the vulnerability to SEU during refresh operation. Even though the refresh operation might not take long time, high ratio of refresh operation in large last level cache makes it significantly vulnerable to SEUs. In Chapter 3, we proposed a new model to estimate the vulnerability of different class of workloads to SEUs while placing in eDRAM-based LLC.

The high write energy overhead and deviation of MTJ resistance due to process variation in STT-MRAM are two primary concerns that must be addressed in pre- and post-fabrication process. In Chapter 5, we propose an efficient block allocation/replacement policy to reduce the high write energy overhead existing in hybrid eDRAM/STT-MRAM cache design. Furthermore, in Chapter 4, we propose a novel SA arrangement for sub-banks of STT-MRAM based LLC to improve sensing reliability while maintaining energy consumption in the power budget.

7.1 Technical Summary

7.1.1 Bit-Upset Vulnerability Factor Analysis for eDRAM-based LLC

The proposed model in Chapter 3 relies on LLC ensemble behavior analysis using trace files obtained from PARSEC benchmark suites running on an extended version of MARSSx86. The technical summary of this work can be listed as following:

- preserving data integrity in eDRAM needs periodic refresh of the charge in each capacitor,
- this new lifetime sequence needs to be investigated to accurately reveal how different lifetime sequences of cache data contribute to vulnerability,
- the proposed model investigates three SER modes for eDRAM cells in which the minimum collected charge has the potential to induce soft error,
- the Memory, bit and bit-bar SER modes are identified and integrated into the proposed model, and
- the experimental results show that the vulnerable sequences account for about 27.24% of data array lifetime in the cache, among which RR contributes about 23.45% to BUVF.

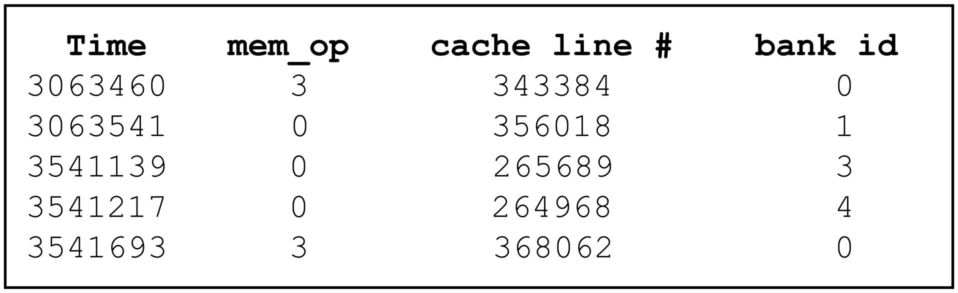
7.1.2 Variation-Immune NVM-based System Design

We implemented SOS presented in Chapter 4 through following steps:

Employ bank access analysis using collected tracefiles for PARSEC benchmarks suite

First, the simulation results were executed on a 1-bit STT-MRAM cell equipped with combination of PCSA and SPCSA using HSPICE to run 10,000 Monte Carlo simulations. To accurately analyze the impact of PV on NVM cell arrays, we run the Monte Carlo simulations on a 32-bit sub-bank that is equipped with 32 conventional SAs. Then, the energy consumption and performance of this model was expanded to simulate a 64-Byte data block. This data was integrated into NVSim to report the total energy consumption and delay of an array of SAs. The extracted results from NVSim were integrated into MARSSX86 later on. Thus, the PV effect is applied into the IPC and total energy consumption of executed PARSEC benchmark suite.

To report the accessed bank id for each cache line in the tracefile, we extended MARSSX86 simulator. This extension includes changes in PTLSim module such as cacheController.cpp, cacheController.h, memoryController.cpp, etc. On the other hand, we modify ptlsim.cpp to add required variables into LLC-access-trace.log which prints the access pattern to each cache line. For example, our generated tracefile was similar to Fig. 7.1 in which the simulation time, memory operation, cache line number, and bank id are listed.



```
  Time     mem_op    cache line #    bank id
3063460      3          343384          0
3063541      0          356018          1
3541139      0          265689          3
3541217      0          264968          4
3541693      3          368062          0
```

Figure 7.1: The modified tracefile to include bank id.

Next, we developed a python-based framework to analyze cache line access pattern in each set of cache as illustrated in Fig. 7.2. Accordingly, we were able to extract the holistic LLC access behavior of the workloads. In particular, the write-intensive and read-intensive cache lines were determined using this script.

```
import sys
import re
import operator
import os
with open('../llc_access_trace.log', 'r') as fin:
        lines = [line.split() for line in fin]
        lines.sort(key=operator.itemgetter(2))
with open('llc_access_trace_sorted.log', 'w') as fout:
    for el in lines:
        fout.write('{0}\n'.format(' '.join(el)))
with open('llc_access_trace_sorted.log', 'r') as ftrace:
 for line in ftrace:
        if int(line.split()[3]) < 0:    // access to Bank 0
            if int(line.split()[1]) == 0:
                readCounterBank0 += 1
            elif int(line.split()[1]) == 1:
                      writeCounterBank0 += 1
            elif int(line.split()[1]) == 3:
                      evictCounterBank0 += 1
```

Figure 7.2: A part of under-develop python parser script to analyze cache line access pattern.

As the next step, we fed the VARIUS [150] tool with the new LLC configuration to extract the PV map for cache bank floorplan of STT-MRAM based LLC. VARIUS generates the die floorplan that can be used to sketch sub-banks region in each bank. In addition, the intensity of PV on different sub-banks has been shown with different colors. It basically helps to reveal the correlation between the impacted cache lines due to PV effect in terms of consumed energy, delivered performance, and offered reliable sensing operation.

Propose and implement reliable and low-energy sense amplifier

In Chapter 4, we introduce a naive approach to utilize a high-reliable SA [85] and a low-power SA [186] which were proposed by other research groups. As an extension to our previous work, we designed and tested a new set of SAs using Mont Carlo Simulation in HSPICE which offers both high-reliable and low-energy consumption. Thus, we can eliminate MUX component from the merges SAs. Next, we compared our proposed SAs with previous SAs designs in terms of reliability and energy consumption. Our results show that our proposed SA can improve both total energy consumption and reliable sensing operation. We applied the results of the proposed SA into NVSim to obtain the new consumed energy and latency. These results integrated into computer-system simulator to examine the impact of new SA utilization in performance delivery of new configuration.

Develop remap algorithms to improve the consumed energy and performance while increasing the dependability of HW architecture

The memory access to the cache lines has a non-uniform pattern. This means that some cache lines in the banks may experience a large number of read operation while others may be accessed by frequent write operation. This non-uniform access can be problematic when the read-intensive cache lines are placed on a high PV-impacted region of a bank. In this case, SOS attempts to utilize high-reliable SA for read sensing operation whereby there is still no guarantee that the sensed data has been read correctly while the consumed energy for this process is significantly high. If we assume that this portion of cache will be frequently accessed, the probability of application's contamination with corrupted data will increase. To reduce the risk of this contamination, we proposed to remap frequently read accessed blocks to low-PV impacted region of banks. This re-organization will reduce the risk of incorrect sensing operation while the total consumed energy will reduce significantly. In addition, we proposed to transfer write-intensive blocks to SRAM data array to amortize the latency and high dynamic energy consumpton associated with incoming write operations.

We executed PARSEC benchmark suite on modified simulator to collect the new tracefiles. Next, we re-analyzed the new tracefiles for extracting cache blocks access pattern in each set by considering the PV impact on each bank. Finally, the energy consumption, performance, and correctly data sensing were re-calculated and compared with the original design.

7.1.3 Accelerate Service to Critical Loads

The proposed technique in Chapter 5 relies on simulator extension and LLC ensemble behavior analysis using trace files obtained from PARSEC benchmark suite. The technical summary of this work can be listed as following:

- to maximize the throughput of hybrid cache designs, novel allocation/replacement policies are required,
- the service to the critical loads must be prioritized over other operations,
- the data movement from LLC to L2 take significant cycle,
- a fraction of working set my exhibit distant re-reference interval that may repeatedly be evicted and be brought back from/to L2, and
- there is a high spatial locality at page-level granularity among different class of workloads.

7.1.4 LI2 Computing: Energy-aware Low-Rank Matrix Data In-transit Reconstruction

The LI2 idea is still in under implementation. However, the following steps are required to realize LI2 computation approach:

Conduct an study regarding the typical access cost to large low-rank sparse matrix resided in memory

At the beginning, we need to develop a test bench which accesses the entries of matrix with three different access behavior (row-access, column-access, diagonal access) and indirect reference access. The tracefiles for these accesses must be collected and analyzed to calculate consumed energy and performance cost due to data movement from off-chip main memory to on-chip CPU. Then, we will be able to formulate the energy cost for data movement in these applications.

Develop the MATLAB code to realize matrix factorization which is the decomposition of the low-rank matrix to two smaller matrices

Modify computer-system simulator to realize reconstruction function of requested column

As mentioned before, we need to add a reconstitution scratchpad component to the cache controller. Next, the LI2 logic fabric module must be created and be connected to other modules in the simulator. In addition, the related files in simulator must be modified to look for missed items in the right component (if the address is in the range for matrix elements, go to reconstitution scratchpad. Otherwise, send the request to main memory). The decomposed matrices must be loaded into scratchpad after finding them and prior to run the test bench. Then, run the test bench on the loaded sparse matrix in main memory and collect the tracefiles.

Re-calculate energy and performance for the modified architecture

Finally, we have to compare the results obtained from LI2 with the results from original architecture. Our preliminary study implies that LI2 has the potential to reduce the energy consumption associated with data transfer by around 20% at 1.5x higher performance than baseline system.

7.2 Technical Insights Gained

By analysis the SEU effect on eDRAM-based LLC presented in Chapter 3, we observed that:

- The vulnerable sequences account for about 27.24% of data array lifetime in the cache, among which RR contributes about 23.45%. The data instances in MRR and LRR sequences are dominate the overall RR time, 23.14% on the average across the entire suite. This observation confirms that the long-term residency of a data block in LLC between two consecutive accesses significantly increases its vulnerability to soft errors. The profile results convince us that the read intensive benchmarks with medium and long read-read instances, i.e. $streamcluster$, contribute the most BUVF in the data cache.

- The results obtained by BUVF characterization and analysis are shown in Fig. 3.9. A smaller value of BUVF implies that the cache is more resilient against soft errors. The BUVF values for $raytrace$ and $facesim$ benchmarks are 0.020 and 0.022, respectively, indicating the high resiliency of them to soft errors. The main reason behind the reduced BUVF value for these benchmarks is that the data remained for a shorter period within the vulnerable sequences. On the other hand, the BUVF value of $streamcluster$ is 0.76 which is the highest BUVF among benchmarks under study.
- Even though the data located in LRR sequence only account for 0.73 percent of total data items as shown in Fig. 3.8 (a), this sequence is the second largest contributor to BUVF. Furthermore, the two potential duration sequences MRR and LRR together account for 0.16 of vulnerability factor across all workloads, pointing out that the RR sequence contributes the most to BUVF. This agrees with typical distribution of data access being read-predominant versus write-perdominant.

By analysis the PV effect on NVM-based LLC presented in Chapter 4, we observed that:

- The PV in MTJ devices utilized in STT-MRAM organization, manifests itself as variation in MgO thickness and MTJ geometry which in turn results in deviation of MTJ resistance,
- the threshold voltage, V_{th}, and gate length, L_{eff}, of CMOS access transistors in STT-MRAM organization exhibit delay and driving current variations under PV, which negatively impacts

the performance consistency of memory operation,

- as the results of PV effect on both conventional and emerging semiconductor technologies, the difference between the sensed bit-line voltage and the reference voltage which is referred as *sense margin* can severely fluctuate, resulting in possible false detection scenario and increased bit error rate,
- our experimental results indicate that the PV effect may incur around 27.5% of the sensed data to be read incorrectly from which 21.5% are extremely vulnerable which implies that around one fifth of the overall sensing operations have the potential to contaminate the application's data structure, and
- the VFDS in the hybrid cache design is reduced by 89% on average compared to LLC with STT-MRAM, thus improving the mean TDS from 72.5% to 97% across all workloads.

By analysis the results of applying presented idea in Chapter 5, we observed that:

- Extensive Read Reused Access (ERRA) blocks and frequently reused blocks with distant re-reference interval in L2 must be prioritized for service,
- more than 50% of read accesses are from one of the three pages with the most accesses in the selected workloads, indicating that the majority of cache block accesses are from the same physical page.,
- RRAP reduces the normalized RST around 1.47% compared to regular STT-MRAM, and
- RRAP reduces the read miss ratio of the regular STT-MRAM based L2 design by 51.74% on average.

7.3 Future Works

As the future work, the ongoing LI2 project will be continued to provide a hardware accelerator for machine learning or deep learning techniques that deal with low-rank matrices. The advantages and disadvantages of LI2 concept in terms of memory bandwidth improvement, energy consumption, IPC improvement will be elaborated in detail.

LIST OF REFERENCES

[1] The r project for statistical computing. *http://www.r-project.org/*.

[2] 2016.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[4] A. Agrawal, A. Ansari, and J. Torrellas. Mosaic: Exploiting the spatial locality of process variation to reduce refresh energy in on-chip edram modules. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, pages 84–95, Feb 2014.

[5] A. Agrawal, P. Jain, A. Ansari, and J. Torrellas. Refrint: Intelligent Refresh to Minimize Power in on-chip Multiprocessor Cache Hierarchies. In *Proceedings of 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 400–411. IEEE, 2013.

[6] M. Ahmadian, A. Paya, and C. D. Marinescu. Security of Applications Involving Multiple Organizations - Order Preserving Encryption in Hybrid Cloud Environments. In *Proceedings of 28th International Parallel and Distributed Processing Symposium Workshops*, pages 894–903. IEEE, 2014.

[7] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi. A scalable processing-in-memory accelerator for parallel graph processing. In *ACM SIGARCH Computer Architecture News*, volume 43, pages 105–117. ACM.

[8] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi. A scalable processing-in-memory accelerator for parallel graph processing. In *ACM SIGARCH Computer Architecture News*, volume 43, pages 105–117. ACM, 2015.

[9] J. Ahn, S. Yoo, O. Mutlu, and K. 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.

[10] J. H. Ahn, M. Erez, and W. J. Dally. Scatter-add in data parallel architectures. In *11th International Symposium on High-Performance Computer Architecture*, pages 132–142. IEEE.

[11] R. Arroyo, R. Harrington, S. Hartman, and T. Nguyen. Ibm power7 systems. *IBM Journal of Research and Development*, 55(3):2–1, 2011.

[12] H. Asadi, V. Sridharan, M. Tahoori, and D. Kaeli. Vulnerability analysis of l2 cache elements to single event upsets. In *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, volume 1, pages 1–6, March 2006.

[13] H. Asghari-Moghaddam, Y. H. Son, J. H. Ahn, and N. S. Kim. Chameleon: Versatile and practical near-dram acceleration architecture for large memory systems. In *Microarchi-*

tecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on, pages 1–13. IEEE, 2016.

[14] R. Balasubramonian, J. Chang, T. Manning, J. H. Moreno, R. Murphy, R. Nair, and S. Swanson. Near-data processing: Insights from a micro-46 workshop. *IEEE Micro*, 34(4):36–42, 2014.

[15] R. Balasubramonian, S. Dwarkadas, and D. H. Albonesi. Reducing the complexity of the register file in dynamic superscalar processors. In *Microarchitecture, 2001. MICRO-34. Proceedings. 34th ACM/IEEE International Symposium on*, pages 237–248. IEEE, 2001.

[16] R. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. In *Device and Materials Reliability, IEEE Transactions on*, volume 5, pages 305–316, Sept 2005.

[17] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. Loh, D. McCauley, P. Morrow, D. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die stacking (3d) microarchitecture. In *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, pages 469–479, Dec 2006.

[18] M. N. Bojnordi and E. Ipek. DESC: Energy-efficient Data Exchange using Synchronized Counters. In *Proceedings of 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 234–246. ACM, 2013.

[19] S. Borkar. Thousand Core Chips: A Technology Perspective. In *Proceedings of 44th Annual Design Automation Conference (DAC)*, pages 746–749, New York, NY, USA, 2007. ACM.

[20] S. Borkar and A. A. Chien. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, 2011.

[21] S. Borkar, T. Karnik, and V. De. Design and reliability challenges in nanometer technologies. In *Proceedings of the 41st annual Design Automation Conference*, pages 75–75. ACM, 2004.

[22] M. Brandalero and A. C. S. Beck. Potential analysis of a superscalar core employing a reconfigurable array for improving instruction-level parallelism. *Design Automation for Embedded Systems*, 20(2):155–169, 2016.

[23] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. Ieee.

[24] B. H. Calhoun, Y. Cao, X. Li, K. Mai, L. T. Pileggi, R. A. Rutenbar, and K. L. Shepard. Digital circuit design challenges and opportunities in the era of nanoscale cmos. *Proceedings of the IEEE*, 96(2):343–365, 2008.

[25] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[26] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[27] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.

[28] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob. Technology Comparison for Large Last-level Caches (L 3 Cs): Low-leakage SRAM, Low Write-energy STT-RAM, and Refresh-optimized eDRAM. In *Proceedings of 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 143–154. IEEE, 2013.

[29] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob. Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pages 143–154, Feb 2013.

[30] X. Chen, N. Khoshavi, J. Zhou, D. Huang, R. DeMara, J. Wang, W. Wen, and Y. Chen. AOS: Adaptive Overwrite Scheme for Energy Efcient MLC STT-RAM Cache. In *Proceedings of 53nd Annual Design Automation Conference (DAC)*. ACM, 2016.

[31] D. Cheng, H. Hsiung, B. Liu, J. Chen, J. Zeng, R. Govindan, and S. K. Gupta. A new march test for process-variation induced delay faults in srams. In *2013 22nd Asian Test Symposium*, pages 115–122. IEEE, 2013.

[32] A. K. Chintaluri. Analysis of defects and fault models in embedded spin transfer torque (stt) mram arrays. 2016.

[33] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley Sons, 2009.

[34] G. Da Cunha Rodrigues, R. N. Calheiros, V. T. Guimaraes, G. L. d. Santos, M. B. de Carvalho, L. Z. Granville, L. M. R. Tarouco, and R. Buyya. Monitoring of cloud computing environments: concepts, solutions, trends, and future directions. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 378–383. ACM.

[35] M. Dean and P. Norman. Heterogeneous chip multiprocessors. 2005.

[36] J. W. Demmel. *Applied numerical linear algebra*. SIAM, 1997.

[37] Y. Deng and W. P. Maly. Interconnect characteristics of 2.5-d system integration scheme. In *Proceedings of the 2001 International Symposium on Physical Design*, ISPD '01, pages 171–175, New York, NY, USA, 2001. ACM.

[38] R. Dmer, W. Chen, and X. Han. Parallel discrete event simulation of transaction level models. In *17th Asia and South Pacific Design Automation Conference*, pages 227–231. IEEE.

[39] M. F. Dolz, F. D. Igual, T. Ludwig, L. Piuel, and E. S. Quintana-Ort. Balancing task-and data-level parallelism to improve performance and energy consumption of matrix computations on the intel xeon phi. *Computers and Electrical Engineering*, 46:95–111, 2015.

[40] M. Dong, Q. Yu, X. Zhou, Y. Hong, H. Chen, and B. Zang. Rethinking benchmarking for nvm-based file systems. In *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*, page 20. ACM, 2016.

[41] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A Circuit-level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(7):994–1007, 2012.

[42] J. Draper, J. Chame, M. Hall, C. Steele, T. Barrett, J. LaCoss, J. Granacki, J. Shin, C. Chen, and C. W. Kang. The architecture of the diva processing-in-memory chip. In *Proceedings of the 16th international conference on Supercomputing*, pages 14–25. ACM.

[43] E. Eken, Y. Zhang, W. Wen, R. Joshi, H. Li, and Y. Chen. A Novel Self-Reference Technique for STT-RAM Read and Write Reliability Enhancement. *IEEE Transactions on Magnetics*, 50(11):1–4, 2014.

[44] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R. McKenzie. Computational ram: implementing processors in memory. *IEEE Design and Test of Computers*, 16(1):32–41, 1999.

[45] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Architecture support for disciplined approximate programming. In *ACM SIGPLAN Notices*, volume 47, pages 301–312. ACM, 2012.

[46] B. Falsafi, M. Stan, K. Skadron, N. Jayasena, Y. Chen, J. Tao, R. Nair, J. Moreno, N. Muralimanohar, and K. Sankaralingam. Near-memory data services. *IEEE Micro*, 36(1):6–13, 2016.

[47] Y.-P. Fang, B. Vaidyanathan, and A. Oates. Soft error rate cross-technology prediction on embedded dram. In *Reliability Physics Symposium, 2009 IEEE International*, pages 925–928, April 2009.

[48] Z. Fang, L. Zhang, J. B. Carter, S. A. McKee, A. Ibrahim, M. A. Parker, and X. Jiang. Active memory controller. *The Journal of Supercomputing*, 62(1):510–549, 2012.

[49] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim. Nda: Near-dram acceleration architecture leveraging commodity dram devices and standard memory modules. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 283–295. IEEE.

[50] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim. Drama: An architecture for accelerated processing near memory. *IEEE Computer Architecture Letters*, 14(1):26–29, 2015.

[51] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim. Drama: An architecture for accelerated processing near memory. *IEEE Computer Architecture Letters*, 14(1):26–29, 2015.

[52] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim. Nda: Near-dram acceleration architecture leveraging commodity dram devices and standard memory modules. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 283–295. IEEE, 2015.

[53] R. Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Pearson Higher Education, 2004.

[54] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.

[55] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. M. Rabaey, and C. J. Spanos. Modeling within-die spatial correlation effects for process-design co-optimization. In *ISQED*, volume 5, pages 516–521. Citeseer, 2005.

[56] B. Furht and A. Escalante. *Handbook of cloud computing*, volume 3. Springer, 2010.

[57] S. Ganapathy, R. Canal, D. Alexandrescu, E. Costenaro, A. Gonzalez, and A. Rubio. A novel variation-tolerant 4t-dram cell with enhanced soft-error tolerance. In *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, pages 472–477, Sept 2012.

[58] Z. Gao, P. Reviriego, Z. Xu, X. Su, M. Zhao, J. Wang, and J. Maestro. Fault tolerant parallel ffts using error correction codes and parseval checks. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99):1–1, 2015.

[59] P. Garcia, K. Rupnow, and K. Compton. A reconfigurable computing scheduler optimized for multicore systems. In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pages 107–112. IEEE, 2010.

[60] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[61] B. Giridhar, M. Cieslak, D. Duggal, R. Dreslinski, H. M. Chen, R. Patti, B. Hold, C. Chakrabarti, T. Mudge, and D. Blaauw. Exploring dram organizations for energy-efficient and resilient exascale memories. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 23. ACM, 2013.

[62] M. Gokhale, B. Holmes, and K. Iobst. Processing in memory: The terasys massively parallel pim array. *Computer*, 28(4):23–31, 1995.

[63] A. González, C. Aliagas, and M. Valero. A data cache with multiple caching strategies tuned to different types of locality. In *ACM International Conference on Supercomputing 25th Anniversary Volume*, pages 217–226. ACM, 2014.

[64] D. Gregorek, R. Schmidt, and A. Garca-Ortiz. Transaction level analysis for a clustered and hardware-enhanced task manager on homogeneous many-core systems. *arXiv preprint arXiv:1502.02852*, 2015.

[65] Q. Guo, X. Guo, Y. Bai, and E. Ipek. A resistive tcam accelerator for data-intensive computing. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 339–350. ACM.

[66] Q. Guo, X. Guo, R. Patel, E. Ipek, and E. G. Friedman. Ac-dimm: associative computing with stt-mram. In *ACM SIGARCH Computer Architecture News*, volume 41, pages 189–200. ACM.

[67] Q. Guo, X. Guo, R. Patel, E. Ipek, and E. G. Friedman. Ac-dimm: associative computing with stt-mram. In *ACM SIGARCH Computer Architecture News*, volume 41, pages 189–200. ACM, 2013.

[68] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[69] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[70] J. Hurwitz, A. Nugent, F. Halper, and M. Kaufman. *Big data for dummies*. John Wiley and Sons, 2013.

[71] M. Imani, P. Mercati, and T. Rosing. Remam: Low energy resistive multi-stage associative memory for energy efficient computing. In *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pages 101–106, March 2016.

[72] M. Imani, S. Patil, and T. Rosing. DCC: Double Capacity Cache for Narrow-Width Data Values. *Great Lakes Symposium on VLSI*, 2016.

[73] M. Imani, S. Patil, and T. Rosing. Low Power Data-Aware STT-RAM based Hybrid Cache Architecture. *Proceedings of 17th International Symposium on Quality Electronic Design (ISQED)*, 2016.

[74] M. Imani, A. Rahimi, Y. Kim, and T. Rosing. A Low-Power Hybrid Magnetic Cache Architecture Exploiting Narrow-Width Values. *Non-Volatile Memory Systems and Applications Symposium*, 2016.

[75] A. Jaleel, M. Mattina, and B. Jacob. Last Level Cache (LLC) Performance of Data Mining Workloads on a CMP-a Case Study of Parallel Bioinformatics Workloads. In *Proceedings of 12th International Symposium on High Performance Computer Architecture (HPCA)*, pages 88–98. IEEE, 2006.

[76] A. Jaleel, K. B. Theobald, S. C. Steely Jr, and J. Emer. High performance cache replacement using re-reference interval prediction (rrip). In *ACM SIGARCH Computer Architecture News*, volume 38, pages 60–71. ACM, 2010.

[77] K. J. Janik, S.-L. L. Lu, and M. F. Miller. Non-stalling circular counterflow pipeline processor with reorder buffer, Dec. 19 2000. US Patent 6,163,839.

[78] L. Jiang, W. Wen, D. Wang, and L. Duan. Improving Read Performance of STT-MRAM based Main Memories through Smash Read and Flexible Read. In *Proceedings of 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 31–36. IEEE, 2016.

[79] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das. Cache Revive: Architecting Volatile STT-RAM Caches for Enhanced Performance in CMPs. In *Proceedings of 49th Annual Design Automation Conference (DAC)*, pages 243–252. ACM, 2012.

[80] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[81] M. R. Jokar, M. Arjomand, and H. Sarbazi-Azad. Sequoia: A high-endurance nvm-based cache architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(3):954–967, 2016.

[82] M. R. Jokar, M. Arjomand, and H. Sarbazi-Azad. Sequoia: A High-Endurance NVM-Based Cache Architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2016.

[83] I. Jolliffe. Principal component analysis: Wiley online library. 2005.

[84] N. P. Jouppi and D. W. Wall. *Available instruction-level parallelism for superscalar and superpipelined machines*, volume 17. ACM, 1989.

[85] W. Kang, E. Deng, J.-O. Klein, Y. Zhang, Y. Zhang, C. Chappert, D. Ravelosona, and W. Zhao. Separated precharge sensing amplifier for deep submicrometer mtj/cmos hybrid logic circuits. *IEEE Transactions on Magnetics*, 50(6):1–5, 2014.

[86] W. Kang, Z. Li, Y. Cheng, J.-O. Klein, Y. Zhang, D. Ravelosona, C. Chappert, and W. Zhao. A dynamic reference scheme to improve the sensing reliability of magnetic random access memory. In *Solid-State and Integrated Circuit Technology (ICSICT), 2014 12th IEEE International Conference on*, pages 1–3. IEEE, 2014.

[87] W. Kang, W. Zhao, Z. Wang, J.-O. Klein, Y. Zhang, D. Chabi, Y. Zhang, D. Ravelosona, and C. Chappert. An Overview of Spin-based Integrated Circuits. In *Proceedings of 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 676–683. IEEE, 2014.

[88] Y. Kang, W. Huang, S. M. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik, and J. Torrellas. Flexram: Toward an advanced intelligent memory system. In *2012 IEEE 30th International Conference on Computer Design (ICCD)*, pages 5–14.

[89] S. Khan, A. R. Alameldeen, C. Wilkerson, O. Mutlu, and D. A. Jimenezz. Improving Cache Performance using Read-write Partitioning. In *Proceedings of 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 452–463. IEEE, 2014.

[90] N. Khoshavi, X. Chen, J. Wang, and R. DeMara. Bit-Upset Vulnerability Factor for eDRAM Last Level Cache Immunity Analysis. In *Proceedings of 17th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2016.

[91] N. Khoshavi, X. Chen, J. Wang, and R. F. DeMara. Bit-upset vulnerability factor for edram last level cache immunity analysis. In *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pages 6–11. IEEE, 2016.

[92] N. Khoshavi, X. Chen, J. Wang, and R. F. DeMara. Read-Tuned STT-RAM and eDRAM Cache Hierarchies for Throughput and Energy Enhancement. In *arXiv preprint arXiv:1607.08086*, 2016.

[93] D. Kim, S. Yoo, and S. Lee. Hybrid Main Memory for High Bandwidth Multi-Core System. *IEEE Transactions on Multi-Scale Computing Systems*, 1(3):138–149, July 2015.

[94] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[95] M. K. Kim, J. H. Choi, J. W. Kwak, S. T. Jhang, and C. S. Jhon. Bypassing method for stt-ram based inclusive last-level cache. In *Proceedings of the 2015 Conference on research in adaptive and convergent systems*, pages 424–429. ACM, 2015.

[96] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carson, W. Dally, M. Denneau, P. Franzon, W. Harrod, and K. Hill. Exascale computing study: Technology challenges in achieving exascale systems. 2008.

[97] M. P. Komalan, C. Tenllado, J. I. G. Pérez, F. T. Fernández, and F. Catthoor. System level exploration of a stt-mram based level 1 data-cache. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 1311–1316. EDA Consortium, 2015.

[98] W. Kong, P. Parries, G. Wang, and S. Iyer. Analysis of retention time distribution of embedded dram - a new method to characterize across-chip threshold voltage variation. In *Test Conference, 2008. ITC 2008. IEEE International*, pages 1–7, Oct 2008.

[99] D. Kuang, S. Yun, and H. Park. Symnmf: nonnegative low-rank approximation of a similarity matrix for graph clustering. *Journal of Global Optimization*, 62(3):545–574, 2015.

[100] A. Kumar and R. Mahapatra. Enhancing tlb reach with ternary-cam cells. Technical report, Technical Report, 2004.

[101] S. Kumar, M. Mohri, and A. Talwalkar. On sampling-based approximate spectral decomposition. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 553–560. ACM, 2009.

[102] G. Kurian, S. Devadas, and O. Khan. Locality-aware Data Replication in the Last-level Cache. In *Proceedings of 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12. IEEE, 2014.

[103] G. Kurian, O. Khan, and S. Devadas. The locality-aware adaptive cache coherence protocol. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 523–534, New York, NY, USA, 2013. ACM.

[104] K. W. Kwon, S. H. Choday, Y. Kim, and K. Roy. Aware (asymmetric write architecture with redundant blocks): A high write speed stt-mram cache architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(4):712–720, April 2014.

[105] H. Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 7(3):1–121, 2014.

[106] J. Li, P. Ndai, A. Goel, S. Salahuddin, and K. Roy. Design Paradigm for Robust Spin-Torque Transfer Magnetic RAM (STT MRAM) From Circuit/Architecture Perspective. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(12):1710–1723, 2010.

[107] X. Liang, R. Canal, G.-Y. Wei, and D. Brooks. Process variation tolerant 3t1d-based cache architectures. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 15–26. IEEE Computer Society, 2007.

[108] C. Lin and J.-N. Chiou. High-endurance hybrid cache design in cmp architecture with cache partitioning and access-aware policies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(10):2149–2161, 2015.

[109] A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Canegallo, and R. Guerrieri. A vliw processor with reconfigurable instruction set for embedded applications. *IEEE Journal of solid-state circuits*, 38(11):1876–1886, 2003.

[110] G. Loh and M. Hill. Supporting very large dram caches with compound-access scheduling and missmap. *Micro, IEEE*, 32(3):70–78, May 2012.

[111] G. H. Loh and M. D. Hill. Efficiently enabling conventional block sizes for very large die-stacked dram caches. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 454–464, New York, NY, USA, 2011. ACM.

[112] G. H. Loh and M. D. Hill. Efficiently Enabling Conventional Block Sizes for Very Large Die-stacked DRAM Caches. In *Proceedings of 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 454–464. ACM, 2011.

[113] D. A. Luick. Multiple parallel pipeline processor having self-repairing capability, Oct. 17 2006. US Patent 7,124,318.

[114] M. Maghsoudloo and H. Zarandi. Dirty data vulnerability mitigation by means of sharing management in cache coherence protocols. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on*, pages 205–210, Oct 2012.

[115] M. Maghsoudloo, H. Zarandi, S. Mozafari, and N. Khoshavi. Soft error detection technique in multi-threaded architectures using control-flow monitoring. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 789–792, Aug 2011.

[116] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz. Smart memories: A modular reconfigurable architecture. In *ACM SIGARCH Computer Architecture News*, volume 28, pages 161–171. ACM.

[117] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[118] M. Maresca and H. Li. Connection autonomy in simd computers: a vlsi implementation. *Journal of Parallel and Distributed Computing*, 7(2):302–320, 1989.

[119] I. Markovsky and K. Usevich. Software for weighted structured low-rank approximation. *Journal of Computational and Applied Mathematics*, 256:278–292, 2014.

[120] S. McFarling. Cache replacement with dynamic exclusion. In *ACM SIGARCH Computer Architecture News*, volume 20, pages 191–200. ACM, 1992.

[121] S. Mittal and J. S. Vetter. Ayush: A technique for extending lifetime of sram-nvm hybrid caches. *IEEE Computer Architecture Letters*, 14(2):115–118, 2015.

[122] D. Nassimi and S. Sahni. Data broadcasting in simd computers. *IEEE Transactions on Computers*, 100(2):101–107, 1981.

[123] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori. Evaluation of hybrid memory technologies using sot-mram for on-chip cache hierarchy. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(3):367–380, 2015.

[124] F. Oboril, F. Hameed, R. Bishnoi, A. Ahari, H. Naeimi, and M. Tahoori. Normally-off stt-mram cache with zero-byte compression for energy efficient last-level caches. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 236–241. ACM, 2016.

[125] W. Oed and O. Lange. On the effective bandwidth of interleaved memories in vector processor systems. *IEEE Transactions on Computers*, 100(10):949–957, 1985.

[126] M. Oskin, F. T. Chong, and T. Sherwood. *Active pages: a computation model for intelligent memory*, volume 26. IEEE Computer Society, 1998.

[127] J. Ou, J. Shu, and Y. Lu. A high performance file system for non-volatile main memory. In *Proceedings of the Eleventh European Conference on Computer Systems*, page 12. ACM, 2016.

[128] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[129] C. Pan, S. Gu, M. Xie, C. Xue, and J. Hu. Wear-Leveling Aware Page Management for Non-Volatile Main Memory on Embedded Systems. *IEEE Transactions on Multi-Scale Computing Systems*, PP(99):1–1, 2016.

[130] A. Patel, F. Afram, and K. Ghose. Marss-x86: A qemu-based micro-architectural and systems simulator for x86 multicore processors. In *1st International Qemu Users Forum*, pages 29–30, 2011.

[131] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons. Text mining using non-negative matrix factorizations. In *SDM*, volume 4, pages 452–456. SIAM, 2004.

[132] G. Pekhimenko, E. Bolotin, M. OConnor, O. Mutlu, T. C. Mowry, and S. W. Keckler. Toggle-aware compression for gpus. *IEEE Computer Architecture Letters*, 14(2):164–168, 2015.

[133] G. Pekhimenko, E. Bolotin, N. Vijaykumar, O. Mutlu, T. C. Mowry, and S. W. Keckler. A case for toggle-aware compression for gpu systems. In *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on*, pages 188–200. IEEE, 2016.

[134] G. Pekhimenko, T. Huberty, R. Cai, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry. Exploiting Compressed Block Size as an Indicator of Future Reuse. In *Proceedings of 21st International Symposium on igh Performance Computer Architecture (HPCA)*, pages 51–63. IEEE, 2015.

[135] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2233–2246, 2012.

[136] M. Poremba, S. Mittal, D. Li, J. S. Vetter, and Y. Xie. DESTINY: A Tool for Modeling Emerging 3D NVM and eDRAM caches. In *Proceedings of 2015 Design, Automation Test in Europe Conference Exhibition*, pages 1543–1546. EDA Consortium, 2015.

[137] I. PRESENT. Cramming more components onto integrated circuits. *Readings in computer architecture*, 56, 2000.

[138] M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely, and J. Emer. Adaptive insertion policies for high performance caching. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 381–391. ACM, 2007.

[139] M. Rahmani and G. Atia. Randomized robust subspace recovery for high dimensional data matrices. *arXiv preprint arXiv:1505.05901*, 2015.

[140] M. Rahmani and G. Atia. A subspace learning approach for high dimensional matrix decomposition with efficient column/row sampling. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1206–1214, 2016.

[141] N. Rathi, A. De, H. Naeimi, and S. Ghosh. Cache bypassing and checkpointing to circumvent data security attacks on sttram. *arXiv preprint arXiv:1603.06227*, 2016.

[142] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[143] P. J. Ribeiro Jr and P. J. Diggle. geor: a package for geostatistical analysis. *R news*, 1(2):14–18, 2001.

[144] S. A. Rooholamin and S. G. Ziavras. Modular vector processor architecture targeting at data-level parallelism. *Microprocessors and Microsystems*, 39(4):237–249, 2015.

[145] S. Roy, S. Chatterjee, C. Giri, and H. Rahaman. Faulty tsvs identification and recovery in 3d stacked ics during pre-bond testing. In *3D Systems Integration Conference (3DIC), 2013 IEEE International*, pages 1–6, Oct 2013.

[146] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Vora. A 45nm 8-core enterprise xeon® processor. In *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, pages 9–12. IEEE, 2009.

[147] M. H. Samavatian, H. Abbasitabar, M. Arjomand, and H. Sarbazi-Azad. An efficient STT-RAM last level cache architecture for GPUs. In *Proceedings of 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2014.

[148] M. H. Samavatian, H. Abbasitabar, M. Arjomand, and H. Sarbazi-Azad. An efficient STT-RAM last level cache architecture for GPUs. In *Proceedings of 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2014.

[149] R. Sampson, M. Yang, S. Wei, C. Chakrabarti, and T. F. Wenisch. Sonic millip3de: A massively parallel 3d-stacked accelerator for 3d ultrasound. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pages 318–329. IEEE.

[150] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. Varius: A model of process variation and resulting timing errors for microarchitects. *IEEE Transactions on Semiconductor Manufacturing*, 21(1):3–13, 2008.

[151] V. Seshadri, S. Yedkar, H. Xin, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry. Mitigating Prefetcher-caused Pollution using Informed Caching Policies for Prefetched Blocks. *ACM Transactions on Architecture and Code Optimization (TACO)*, 11(4):51, 2015.

[152] J. P. Shen and M. H. Lipasti. *Modern processor design: fundamentals of superscalar processors*. Waveland Press, 2013.

[153] H. Shin. Modeling of alpha-particle-induced soft error rate in dram. volume 46, pages 1850–1857, 1999.

[154] J. Sim, J. Lee, M. K. Qureshi, and H. Kim. Flexclusion: balancing cache capacity and on-chip bandwidth via flexible exclusion. In *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, pages 321–332. IEEE, 2012.

[155] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson. Null convention multiply and accumulate unit with conditional rounding, scaling, and saturation. *Journal of Systems Architecture*, 47(12):977–998, 2002.

[156] A. M.-C. So and Y. Ye. Theory of semidefinite programming for sensor network localization. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 405–414. Society for Industrial and Applied Mathematics, 2005.

[157] R. L. Stamps, S. Breitkreutz, J. kerman, A. V. Chumak, Y. Otani, G. E. Bauer, J.-U. Thiele, M. Bowen, S. A. Majetich, and M. Klui. The 2014 magnetism roadmap. *Journal of Physics D: Applied Physics*, 47(33):333001, 2014.

[158] J. Suh, M. Manoochehri, M. Annavaram, and M. Dubois. Soft error benchmarking of l2 caches with parma. volume 39, pages 85–96, 2011.

[159] Z. Sun, X. Bi, and H. Li. Process variation aware data management for stt-ram cache design. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '12, pages 179–184, New York, NY, USA, 2012. ACM.

[160] Z. Sun, X. Bi, and H. Li. Process variation aware data management for stt-ram cache design. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 179–184. ACM, 2012.

[161] Z. Sun, X. Bi, H. H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu. Multi Retention Level STT-RAM Cache Designs with a Dynamic Refresh Scheme. In *Proceedings of 44th annual IEEE/ACM International Symposium on Microarchitecture*, pages 329–338. ACM, 2011.

[162] M. B. Taylor. A landscape of the new dark silicon design regime. *IEEE Micro*, 33(5):8–19, 2013.

[163] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 363–374. IEEE Computer Society, 2008.

[164] N. Tuck and D. M. Tullsen. Initial observations of the simultaneous multithreading Pentium 4 processor. In *Proceedings of 12th International Conference on Parallel Architectures and Compilation Techniques*, pages 26–34. IEEE, 2003.

[165] A. N. Udipi, N. Muralimanohar, and R. Balasubramonian. Non-uniform Power Access in Large Caches with Low-swing Wires. In *Proceedings of International Conference on High Performance Computing (HiPC)*, pages 59–68. IEEE, 2009.

[166] A. Valero, J. Sahuquillo, P. Lopez, and J. Duato. Design of Hybrid Second-Level Caches. *IEEE Transactions on Computers*, 64(7):1884–1897, 2015.

[167] R. Wang, L. Jiang, Y. Zhang, L. Wang, and J. Yang. Selective restore: An energy efficient read disturbance mitigation scheme for future stt-mram. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 21:1–21:6, New York, NY, USA, 2015. ACM.

[168] S. Wang, J. Hu, and S. Ziavras. On the characterization and optimization of on-chip cache reliability against soft errors. volume 58, pages 1171–1184, Sept 2009.

[169] Z. Wang, D. A. Jiménez, C. Xu, G. Sun, and Y. Xie. Adaptive Placement and Migration Policy for an STT-RAM-based Hybrid Cache. In *Proceedings of 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 13–24. IEEE, 2014.

[170] M. Watanabe, N. Niioka, T. Kobayashi, R. Karel, M.-A. Fukase, M. Imai, and A. Kurokawa. An effective model for evaluating vertical propagation delay in tsv-based 3-d ics. In *Quality Electronic Design (ISQED), 2015 16th International Symposium on*, pages 519–523, March 2015.

[171] M. Wei, M. Snir, J. Torrellas, and R. B. Tremaine. A near-memory processor for vector, streaming and bit manipulation workloads. 2005.

[172] U. C. Weiser, D. Perlmutter, and Y. Yaari. Pipeline system for executing predicted branch target instruction in a cycle concurrently with the execution of branch instruction, Nov. 23 1993. US Patent 5,265,213.

[173] D. Worledge, G. Hu, D. W. Abraham, J. Sun, P. Trouilloud, J. Nowak, S. Brown, M. Gaidis, E. O'Sullivan, and R. Robertazzi. Spin Torque Switching of Perpendicular Ta form. *Applied Physics Letters*, 98(2):2501, 2011.

[174] H. Xu, Y. Alkabani, R. Melhem, and A. Jones. FusedCache: A Naturally Inclusive, Racetrack Memory, Dual-Level Private Cache. *IEEE Transactions on Multi-Scale Computing Systems*, PP(99):1–1, 2016.

[175] J. Yang and et al. Radiation-induced soft error analysis of stt-ram: A device to circuit approach. In *Computer-aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2015.

[176] A. Yazdanbakhsh, G. Pekhimenko, B. Thwaites, H. Esmaeilzadeh, O. Mutlu, and T. C. Mowry. Rfvp: Rollback-free value prediction with safe-to-approximate loads. *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(4):62, 2016.

[177] S. Yazdanshenas, M. R. Pirbasti, M. Fazeli, and A. Patooghy. Coding last level stt-ram cache for high endurance and low power. *IEEE computer architecture letters*, 13(2):73–76, 2014.

[178] P. Yiapanis, G. Brown, and M. Lujn. Compiler-driven software speculation for thread-level parallelism. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 38(2):5, 2016.

[179] S. F. Yitbarek, T. Yang, R. Das, and T. Austin. Exploring specialized near-memory processing for data intensive operations. In *2016 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1449–1452. IEEE.

[180] M. K. Yoon, K. Kim, S. Lee, W. W. Ro, and M. Annavaram. Virtual thread: Maximizing thread-level parallelism beyond gpu scheduling limit. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 609–621. IEEE.

[181] C. Zhang, G. Sun, P. Li, T. Wang, D. Niu, and Y. Chen. Sbac: a statistics based cache bypassing method for asymmetric-access caches. In *Proceedings of the 2014 international symposium on Low power electronics and design*, pages 345–350. ACM, 2014.

[182] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski. Top-pim: throughput-oriented programmable processing in memory. In *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, pages 85–98. ACM.

[183] M. Zhang and K. Asanovic. Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors. In *ACM SIGARCH Computer Architecture News*, volume 33, pages 336–345. IEEE Computer Society.

[184] Y. Zhang, I. Bayram, Y. Wang, H. Li, and Y. Chen. Adams: asymmetric differential stt-ram cell structure for reliable and high-performance applications. In *Proceedings of the International Conference on Computer-Aided Design*, pages 9–16. IEEE Press, 2013.

[185] B. Zhao, Y. Du, J. Yang, and Y. Zhang. Process variation-aware nonuniform cache management in a 3d die-stacked multicore processor. *IEEE Transactions on Computers*, 62(11):2252–2265, Nov 2013.

[186] W. Zhao, C. Chappert, V. Javerliac, and J.-P. Noziere. High speed, high stability and low power sensing amplifier for mtj/cmos hybrid logic circuits. *IEEE Transactions on Magnetics*, 45(10):3784–3787, 2009.

[187] W. Zhao, Y. Zhang, T. Devolder, J.-O. Klein, D. Ravelosona, C. Chappert, and P. Mazoyer. Failure and Reliability Analysis of STT-MRAM. *Microelectronics Reliability*, 52(9):1848–1852, 2012.

[188] Q. Zhu, B. Akin, H. E. Sumbul, F. Sadi, J. C. Hoe, L. Pileggi, and F. Franchetti. A 3d-stacked logic-in-memory accelerator for application-specific data intensive computing. In *3D Systems Integration Conference (3DIC), 2013 IEEE International*, pages 1–7. IEEE.