Improving Performance and Endurance for Crossbar Resistive Memory

by

Wen Wen

B.E., Southeast University, 2011

M.E., Southeast University, 2014

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Wen Wen

It was defended on

June 19, 2020

and approved by

Jun Yang, Ph.D., Professor, Department of Electrical and Computer Engineering

Youtao Zhang, Ph.D., Professor, Department of Computer Science

Natasa Miskov-Zivanov, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Jingtong Hu, Ph.D., Associate Professor, Department of Electrical and Computer Engineering

Feng Xiong, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Daqing He, Ph.D., Professor, Department of Informatics and Networked Systems

Dissertation Advisors: Jun Yang, Ph.D., Professor, Department of Electrical and Computer Engineering,

Youtao Zhang, Ph.D., Professor, Department of Computer Science

Improving Performance and Endurance for Crossbar Resistive Memory

Wen Wen, PhD

University of Pittsburgh, 2020

Resistive Memory (ReRAM) has emerged as a promising non-volatile memory technology that may replace a significant portion of DRAM in future computer systems. When adopting crossbar architecture, ReRAM cell can achieve the smallest theoretical size in fabrication, ideally for constructing dense memory with large capacity. However, crossbar cell structure suffers from severe performance and endurance degradations, which come from large voltage drops on long wires.

In this dissertation, I first study the correlation between the ReRAM cell switching latency and the number of cells in low resistant state (LRS) along bitlines, and propose to dynamically speed up write operations based on bitline data patterns. By leveraging the intrinsic in-memory processing capability of ReRAM crossbars, a low overhead runtime profiler that effectively tracks the data patterns in different bitlines is proposed. To achieve further write latency reduction, data compression and row address dependent memory data layout are employed to reduce the numbers of LRS cells on bitlines. Moreover, two optimization techniques are presented to mitigate energy overhead brought by bitline data patterns tracking.

Second, I propose XWL, a novel table-based wear leveling scheme for ReRAM crossbars and study the correlation between write endurance and voltage stress in ReRAM crossbars. By estimating and tracking the effective write stress to different rows at runtime, XWL chooses the ones that are stressed the most to mitigate.

Additionally, two extended scenarios are further examined for the performance and endurance issues in neural network accelerators as well as 3D vertical ReRAM (3D-VRAM) arrays. For the ReRAM crossbar-based accelerators, by exploiting the wearing out mechanism of ReRAM cell, a novel comprehensive framework, ReNEW, is proposed to enhance the lifetime of the ReRAM crossbar-based accelerators, particularly for neural network training. To reduce the write latency in 3D-VRAM arrays, a collection of techniques, including an

in-memory data encoding scheme, a data pattern estimator for assessing cell resistance distributions, and a write time reduction scheme that opportunistically reduces RESET latency with runtime data patterns, are devised.

Table of Contents

List of Tables

List of Figures

Preface

Nearing the end of my journey to a Ph.D. degree, I would like to take this opportunity to thank many fantastic people who I am honored and blessed to work and share my life with in the past wonderful years. First and foremost, I am sincerely grateful to my advisor Professor Jun Yang and my co-advisor Professor Youtao Zhang for their guidance, support and trust. They are the great role model to me as professional, passionate, motivated as well as creative researchers. I really enjoy all discussions and brainstorming in developing new ideas with them, and appreciate their patience, advice and tremendous efforts in helping me innovate and explore in my research. It has always been my greatest pleasure and fortune to have them as my advisors.

I would also like to thank Professor Natasa Miskov-Zivanov, Professor Jingtong Hu, Professor Feng Xiong and Professor Daqing He for serving on my Ph.D. dissertation committee. Their feedback and comments are very constructive and valuable to me in shaping this dissertation.

Further, I want to thank all of my lab mates for receiving their continuous support and help. With them along with me in this journey, I have so many enjoyable and memorable moments inside and outside research.

Last but not the least, I would like to dedicate this dissertation to my family, and express my deepest appreciation for their endless love and support. Sincere thanks to my beloved parents, who brought me up to be a better person and always love me unconditionally. Special thanks to my wife, who is always by my side and encourages me throughout this journey.

1.0 Introduction

Due to increasing demand for large capacity memory in modern data-intensive applications, DRAM, the *de facto* memory technology for constructing main memory, faces severe high leakage power, short refreshing interval, low density and yield issues [44]. Recent studies have proposed to construct future large capacity main memory using emerging non-volatile memory (NVM) technologies, e.g., PCM (Phase Change Memory) [125, 75, 52, 74, 23], STT-MRAM (Spin Transfer Torque Magnetic RAM) [50, 37, 108, 1], and ReRAM (Resistive Memory) [101, 105, 121, 115, 54, 114, 35, 77, 119, 97, 120, 63]. These memory technologies have good scalability, high density, almost zero low leakage power as well as non-volatility characteristics.

Among different NVM technologies, ReRAM has become one of the most promising candidates. ReRAM explores the different resistance states of vertically stacked metal and oxide layers to store information. Comparing to other NVM technologies, ReRAM has better write performance than PCM [111, 38] and better density and scalability than STT-MRAM [37, 65, 96]. When adopting crossbar architecture, ReRAM can achieve the smallest $4F^2$ planar cell size [105]. Moreover, the intrinsic analog current accumulation feature of ReRAM crossbars further propels the popularity of studies on this crossbar array structure for accelerating dot-product calculations between matrices and vectors in neural network computing.

1.1 The Challenges in Deployment of ReRAM Crossbars

1.1.1 Write Performance Bottleneck

With the benefits in density, capacity, non-volatility and small leakage power, however, ReRAM crossbars suffer from large sneaky currents [124, 33, 105, 114, 43]. When performing ReRAM accesses, in particular, RESET operations, we cannot ignore the leakage currents

flowing through half-selected cells on the selected wordline and bitlines. This is because crossbar arrays, even after adopting diode selectors, cannot completely isolate the to-be-written cells from other cells on the selected wordline and bitlines. The large sneak currents not only reduce energy efficiency, but also cause large IR drop on long wires [82], leading to degraded performance and operation reliability. With fast technology scaling, the IR drop issue tends to worsen due to increased wire resistance and array sizes. To ensure operation reliability, ReRAM write operations conservatively use the worst-case access latency of all cells in ReRAM arrays, which leads to significant performance degradation and dynamic energy waste.

1.1.2 Limited Write Endurance

According to [115], ReRAM suffers from unsatisfactory write endurance. Recent studies showed that the endurances of ReRAM chips adopting different resistive materials range from 10^3 to 10^9 [102]. Furthermore, prior studies [14, 34, 69] showed that programming ReRAM cells with longer than necessary pulse length over-SETs or over-RESETs the corresponding cells, leading to orders of magnitude degradation in ReRAM cell lifetime [14]. While optimized write strategies [105, 114] write different rows using different write latencies, the rows being close to the write drivers still get stressed more than others. Adopting traditional wear leveling techniques that evenly distribute writes across all rows in ReRAM space would become less effective — the rows that close to the drivers are approaching their lifetime while others may still have a lot of endurance to use. Thus, it is important to devise a wear leveling approach that considers the stress difference at runtime.

1.1.3 Lifetime Issue in ReRAM Crossbar Based In-memory Computing

In recent years, the neural networks have gained increasing attentions and been successfully applied to a wide range of applications [48, 87, 45, 47]. The increasingly growth in the size of datasets and the number of layers in neural networks help to achieve a better prediction accuracy, but also result in dramatically increased computations and expensive data movement from off-chip memory. Conventional CMOS-based general purpose processors

such as multi-core CPU [91] and GPGPU [48], or specialized hardware accelerators, such as FPGA [113] and ASIC designs [15, 41], are intensively studied and proposed with software and hardware optimizations for neural network applications, however, they still suffer from the large energy consumption and limited memory bandwidth [16].

To address these issues, resistive memory (ReRAM), with adopting crossbar array structure, is proposed to implement dot-product calculations by leveraging its analog current accumulation feature [32, 81, 16, 85, 9, 116]. ReRAM crossbars are able to accelerate neural networks computation with low energy consumption and minimized data movement [81], since they have almost zero leakage power and intrinsically support the processing-in-memory (PIM) computation paradigm.

Though ReRAM crossbar based neural network accelerators own these advantages over conventional CMOS-based accelerators, due to the limited cell endurance [102, 115, 9, 116], they suffer from short programming cycles as weight data stored in ReRAM cells are frequently updated during the neural network training. The write endurance of ReRAM chips can range from 10^6 to 10^{12} [72, 69, 9] with adopting various resistive materials and different programming schemes. On the other hand, training the state-of-the-art deep neural networks usually demands at least 5 orders of magnitudes of weight updates, which essentially leads to frequent ReRAM cell programming. Therefore, enhancing the lifetime of ReRAM crossbars is the key to facilitate its widespread adoption as hardware accelerators for neural network training.

Conventional wear-leveling techniques for NVM (non-volatile memory) based main memory have been well-studied, mostly with a focus on evenly distributing write requests across pages [97, 123, 117, 74]. With distinct programming patterns, ReRAM crossbar based neural network accelerators may potentially demand for an innovative approach. Prior efforts on extending ReRAM crossbar based neural network accelerators either manage to squeeze the endurance of the degraded MLC ReRAM cells [116] or exploit the gradient sparsification and regularly perform row-swapping [116]. However, in order to further improve the endurance of ReRAM crossbars for neural network training, it is necessary to investigate optimal programming strategies by exploiting the mechanism of endurance degradation in

ReRAM crossbars, while taking characteristics of the target application, i.e., neural network training, as well as crossbar array features into account.

1.1.4 Write Performance Issue in 3D Vertical ReRAM

With the fast advances in 3D integration technologies, recent studies [107, 104, 94, 39, 129] have demonstrated that 3D stacking is a viable solution for further improving the bit density of ReRAM arrays. In addition to low energy-efficiency and non-volatility, ReRAM (Resistive Memory) achieves excellent density and scalability by vertically stacking multiple layers of cross-point arrays. With different 3D integration processes, 3D Horizontal ReRAM (3D-HRAM) and 3D Vertical ReRAM (3D-VRAM) are two typical 3D stacked ReRAM array architectures [104, 12]. In recent studies, the 3D-VRAM is more widely adopted for high density memories due to lower fabrication cost [107, 39]. In this dissertation, 3D-VRAM is chosen as the baseline.

Though 3D-VRAM arrays can be used to construct terabit-scale memories [39], similar to 2D ReRAM crossbars, they face severe sneak current issues. A recent work [107] shows that the access voltage degradation on the selected cells tends to worsen with more stacked layers in 3D-VRAM arrays. Most prior work on 3D-VRAM arrays studied device characteristics, circuit modeling, architectural design explorations with assuming the worst-case scenario in arrays [107, 104, 39, 12]. An optimization for write latency of 3D-VRAM is still lacking.

1.2 Research Overview

In order to overcome the challenges summarized above, it is necessary to propose comprehensive architectural solutions based on simulations and modeling across multiple levels, i.e., device, circuitry, architecture and application.

In this dissertation, I focus on mitigating the performance degradation from IR drop [99, 100], and also propose a novel wear leveling scheme for addressing the limited write endurance

of ReRAM crossbars [97], by exploiting in-memory data patterns and without incurring significant overhead. I then extend my study to two scenarios: (1) adopting ReRAM crossbars for accelerating neural network computing [98], limited lifetime issue of ReRAM crossbars is critical, particularly when performing the training task; and (2) in 3D-VRAM arrays, write performance and reliability are dramatically degraded by enormous amount of sneaky paths. Therefore, this dissertation proposes a collection of techniques to address these issues, which are summarized in Table 1.

Table 1: A brief summary of proposed schemes in the dissertation.

Chapter	Proposed Scheme	Challenge	Application
Chapter 5	Data Pattern Profiling & Optimizations	Long Write Latency for ReRAM Crossbars	Main Memory
Chapter 6	Wear Leveling `XWL`	Limited Write Endurance	Main Memory
Chapter 7	Framework `ReNEW`	Limited Lifetime During Neural Network Training	Accelerators
Chapter 8	Data Encoding & Safe- and Aggressive-Data Pattern Estimation	Long Write Latency for 3D-VRAM arrays	Main Memory

1.3 Contributions

1.3.1 Speeding Up RESET Operation

First, I focus on mitigating the performance degradation from IR drop. This part of my work has been published in [100]. My contributions are summarized as follows.

- I study the correlation between the RESET latency of an ReRAM row and the number of the cells in low resistance state (LRS) on selected bitlines. I propose to dynamically

speed up the RESET operations when there are small numbers of LRS cells. Further performance improvement is achieved from exploiting data compression and row address dependent data layout.

- I propose a novel profiling technique to dynamically track the number of LRS cells along different bitlines in the crossbar. By leveraging the in-memory processing capability of ReRAM crossbar, the number of LRS cells in bitlines is periodically detected using current aggregation, an operation having fast speed (comparable to READ operation) and low hardware and performance overheads.
- I propose two profiling optimization techniques, i.e., *selective profiling* and *fine-grained profiling*, to mitigate the energy overhead during profiling. They choose a subset of mats or wordlines to profile so that fewer cells are activated during a profiling operation.
- I evaluate the proposed design and compare it to the state-of-the-art. The experimental results reveal that, my design improves system performance by 20.5% and 14.2%, and reduces memory dynamic energy by 20.3% and 12.6%, compared to the baseline and the state-of-the-art crossbar designs, respectively.

1.3.2 Improving Write Endurance

Second, I propose XWL, a novel table based wear leveling design for addressing the write endurance degradation from IR drop in ReRAM crossbars. This part of my work has been published in [97]. I summarize my contributions as follows.

- I study write endurance variation in ReRAM crossbar, which reveals that the effective write, i.e., the actual degree of ReRAM wearing out, depends on data patterns and row addresses at runtime. To the best of my knowledge, this is the first study revealing the unique wearing characteristic in ReRAM crossbars.
- I propose XWL, a novel table based wear leveling design that tracks the effective writes at runtime. XWL periodically remaps the ReRAM rows that are stressed the most, rather than the ones accumulating the most write counts.
- I evaluate the proposed wear leveling scheme. The experimental results reveal that, my design improves write endurance by 324%, compared to the baseline design.

1.3.3 Enhancing Lifetime for ReRAM Crossbar Based Neural Network Accelerators

Third, I propose to enhance lifetime for ReRAM crossbar based neural network accelerators. To achieve this, a comprehensive framework, ReREW, which consists of techniques that can effectively prolong ReRAM crossbar lifetime during neural network training, is proposed. This part of my work has been published in [98]. A summary of main contributions is listed as follows.

- Unlike many of prior studies, I propose to program ReRAM cells in crossbars in SLC (Single Level Cell) mode for neural network training and in MLC (Multi-Level Cell) mode during the inference, in order to fully take the advantage of longer endurance of SLC ReRAM cells during the training and larger capacity of MLC ReRAM cells for the inference.
- Prior studies show that different in-memory data patterns lead to discrepancies in programming latency and voltage stress, which further causes the disparity of actual wearing out degrees of ReRAM cells. Based on this observation, the optimal programming latency is adopted and an optimized order to update weights, which can maximize the lifetime of ReRAM crossbars, is proposed.
- I analyze the trade-off between endurance and programming conditions, and then present an endurance analytical model for ReRAM cell in SLC mode with different programming strengths. In addition, an analytical study of the trade-off between programming latency and switching probability is presented. Based on these analyses along with the intrinsic error-tolerance of neural network training, I propose to intentionally shorten the programming time to enhance lifetime of ReRAM crossbars at a cost of possibly unsuccessful ReRAM cell switching.
- Inspired by a conventional wear-leveling technique for NVM based main memory, I also propose to shift and update a group of columns between training iterations, which can effectively spread out writes across the whole crossbar.

- Experimental evaluations prove that my proposed techniques reduce the total effective writes to ReRAM crossbar-based accelerators by up to 500.3×, 50.0×, 2.83× and 1.60× over two MLC baselines, SLC baseline and SLC design with optimal timing respectively.

1.3.4 Accelerating 3D Vertical Resistive Memories with Opportunistic Write Latency Reduction

Lastly, I aim to improve write performance in 3D-VRAM arrays by exploring and addressing of the unique issues for 3D vertical ReRAM array architectures. The main contributions are summarized as follows:

- A thorough study of how runtime data patterns stored in vertical layers influence write latency in 3D-VRAM array architectures is presented. In particular, I observe that the number of LRS cells in the selected word-line plane electrode plays an more important role on RESET latency, which is significantly different from that in planar crossbars.
- Two different approaches, i.e., safe and aggressive RESET time estimation schemes, are proposed to optimize RESET latency under the premise of successful switching, based on the runtime estimation of data patterns in a 3D-VRAM array. The aggressive-RESET-time-estimation scheme optimizes the latency to the greatest extent but has a low possibility to conduct a second-round RESET, while the safe-estimation scheme guarantees to switch cells successfully in one round.
- The proposed write schemes are experimentally evaluated and results show that, on average, my proposed design achieves 25.98× write latency reduction, 6.92× performance improvement and 52.4% dynamic energy consumption reduction compared to the baseline.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows. The ReRAM fundamentals are introduced in Chapter 2. Chapter 3 discusses the prior art. In Chapter 4, I build the

ReRAM crossbar circuit model to study the correlation between the RESET latency of an ReRAM row and the number of the cells in low resistance state (LRS) on selected bitlines, as well as write endurance variation in ReRAM crossbar. I elaborate the proposed profiling technique, which can dynamically track the number of LRS cells along different bitlines in the crossbar, to speed up the RESET operations when there are small numbers of LRS cells in Chapter 5. In Chapter 6, XWL, a novel table based wear leveling design that tracks the effective writes at runtime, is proposed. The proposed designs of enhancing lifetime for ReRAM crossbar-based neural network accelerators and the designs for accelerating 3D-VRAM arrays are presented in Chapter 7 and Chapter 8, respectively. Chapter 9 concludes the dissertation.

2.0 Preliminaries

In this chapter, ReRAM fundamentals are discussed. In addition, the sneak current and IR drop issues in ReRAM crossbars are briefly introduced.

2.1 ReRAM Cell Structure

ReRAM is a promising non-volatile memory technology that stores data using cell resistance. As shown in Figure 1, an ReRAM cell is composed of two metal layers on the top and bottom, which are separated by metal oxide layer. Prior study [69] has shown that various metal oxide and electrode materials, such as $CuTe_x/HfO_2$ and $CuTe_x/Al_2O_3$, which have different characteristics such as endurance, retention and scalability, can be used to construct ReRAM cell arrays.

ReRAM is a passive resistive based non-volatile memory technology, which uses different resistance states to represent data values. An ReRAM cell has two legal resistance states: a low resistance state (LRS) to represent logic '1' and a high resistance state (HRS) to represent logical '0'.

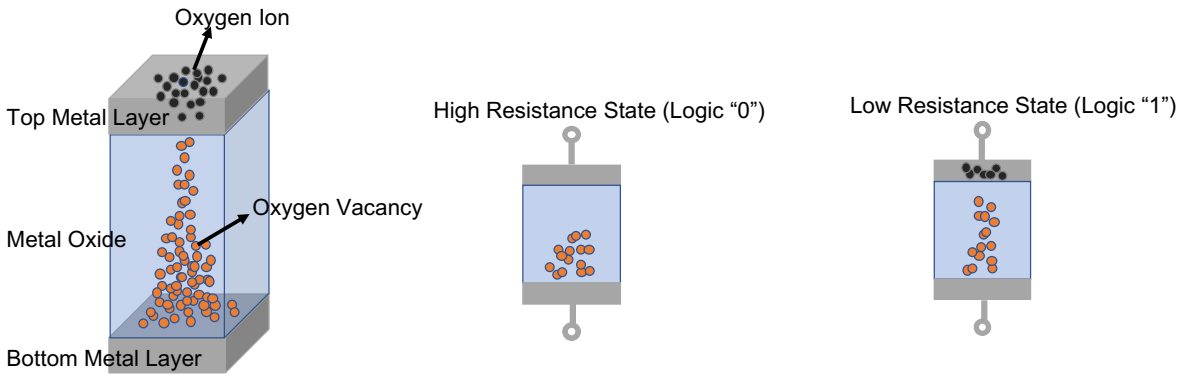


Figure 1: The ReRAM cell structure and two resistance states.

2.2 ReRAM Programming

To program an ReRAM cell (i.e., to switch resistance state from one to the other), a proper voltage with required pulse width and magnitude has to be applied across the cell. Figure 2 depicts two basic programming procedures for ReRAM — RESET and SET, which are reversible switching operations and used to store data in an ReRAM cell. The RESET operation switches the resistance state from LRS to HRS while the SET operation switches from HRS to LRS. For an SLC ReRAM cell, with a positive voltage larger than a certain threshold applied to the top electrode, the current flowing through cell enables a formation of the conductive filaments (CF) in the metal oxide layer, switching the ReRAM cell to low resistance state (LRS). On the contrary, during the RESET process, which is initialized with a negative voltage on the top electrode, the CFs are ruptured and consequently the cell is switched to high resistance state (HRS). To program an MLC ReRAM cell is much more complicated with consuming significantly more power and time [72, 106] and thereby wears out cells much faster, since an iterative programming, i.e., Program & Verify (P&V), is used to accurately achieve the intermediate resistance levels.

2.3 ReRAM Crossbar Array Structure

Figure 3 presents three typical ReRAM array structures. ReRAM array can be fabricated as a grid of 1T1R cells, which is similar to conventional DRAM architecture where each cell is accessed through a transistor. 1T1R cell array has large cell size. ReRAM array can also be organized as a crossbar[1], which achieves the smallest $4F^2$ planar cell size. ReRAM crossbar has low fabrication cost and better scalability and thus is ideal to be architected as DRAM replacement for building large capacity memory.

ReRAM crossbars, depending on if there is a diode access selector, can be categorized as 0T1R or 1D1R structures. Adopting selector helps to reduce sneak currents in the crossbar,

[1]It is also known as cross-point array structure [93].

which enables the fabrication of large cell arrays. In this work, 1D1R crossbar is chosen as the baseline for 2D ReRAM crossbars.

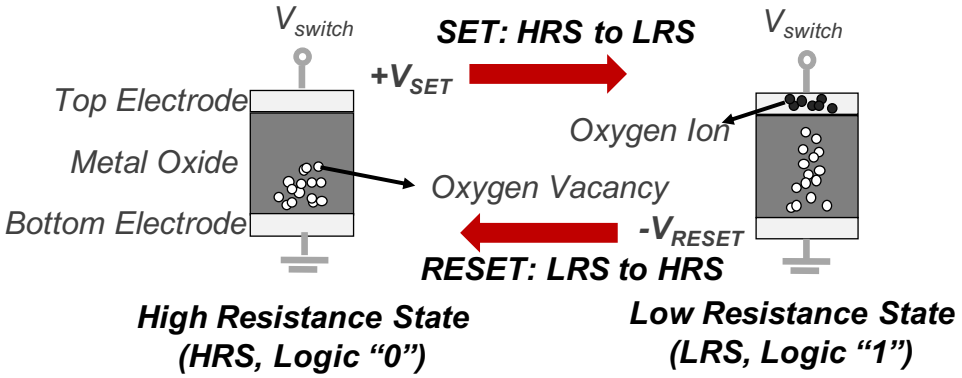


Figure 2: ReRAM cell structure and basic (SET/RESET) programming operations.

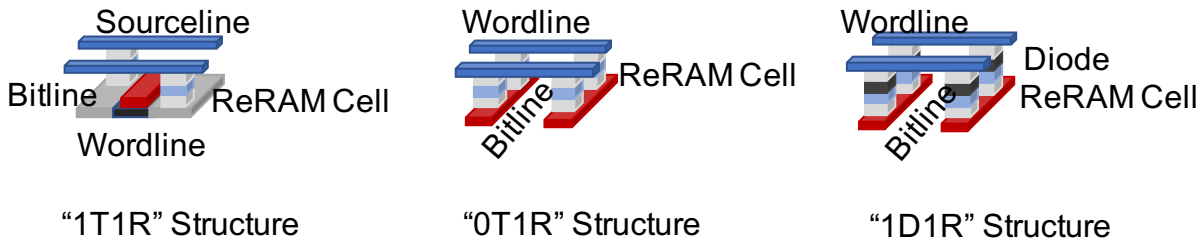


Figure 3: The three typical ReRAM array structures.

2.4 IR Drop Issue

I next study the sneak currents in the crossbar, and will analyze its impact on ReRAM RESET latency in a later chapter.

For discussion purpose, a cacheline is assumed to have 64B and its 512 bits are saved in 64 mats (subarrays) with each subarray containing 8 bits, the same as that in [105]. These mats

spread across 8 chips in one rank. To perform a RESET operation in an ReRAM crossbar, the write driver selects one wordline and up to eight bitlines. The selected wordline is applied with V_{RESET} voltage while each selected bitline is set to 0V. All other bitlines and wordlines are applied with $V_{RESET}/2$. Performing a SET operation is similar but uses opposite current direction. During the write operation, the cells in each subarray can be categorized into three types, as shown in Figure 4 and 5.

- **Selected cells.** They are the cells to be SET or RESET. A selected cell stays on the selected wordline and one of the selected bitlines as well. Ideally they are under the maximal voltage stress, i.e., V_{RESET}.
- **Half-selected cells.** They are the cells on either the selected wordline or the selected bitlines, but not both. Ideally they are under half of the maximal voltage stress, i.e., $V_{RESET}/2$.
- **Not-selected cells.** They are the rest of the cells in the crossbar. Ideally they have no voltage stress.

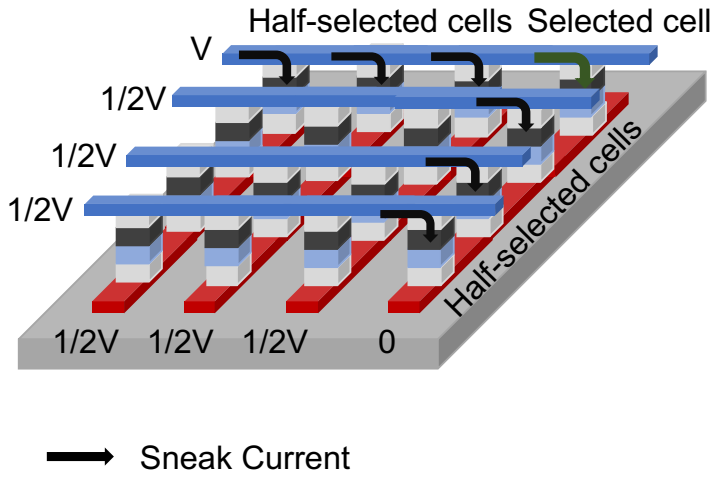


Figure 4: The IR drop issue in ReRAM crossbar array.

A cacheline write operation consists of two phases: a RESET phase to write all 0s and a SET phase to write all 1s. The *DSGB* is adopted to improve write performance [105] and *flip-n-write* is employed to only write modified cells [17]. Based on my experiments as

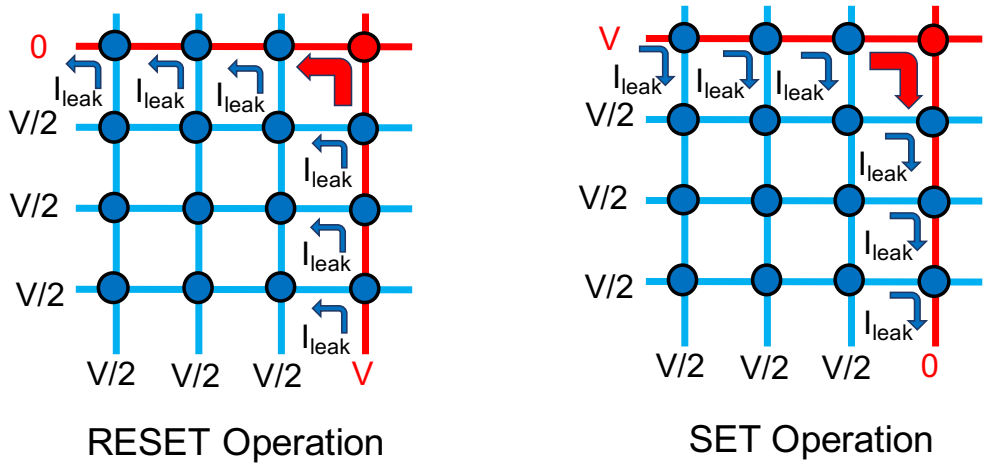


Figure 5: The sneaky currents during RESET and SET operations.

well as prior studies [105, 121, 114], SET operation takes much shorter time than RESET operation, making it less sensitive to voltage stress degradation. Therefore, I focus on long latency RESET operations in the dissertation. The proposed scheme is applicable to the ReRAM structures that have comparable SET and RESET latencies.

Studies have shown that ReRAM crossbar, even adopting diode selectors, has the currents flowing through all cells — while the sneaky currents flowing through not-selected cells are negligible, those flowing through half-selected cells are not. The sneak currents introduce large voltage drop along the wordline and bitlines, referred to as *IR drop* in the crossbar. Large IR drop not only hurts the energy efficiency, but also degrades the performances and write reliability. A recent study has shown that, due to IR drop, it takes longer time to RESET the ReRAM rows that are far away from the write driver [114].

With fast technology scaling, future ReRAM chips are expected to build upon large ReRAM mats, i.e., crossbars. Unfortunately, large crossbars have large wire resistance, which worsens the IR drop issue.

3.0 Prior Art

In this chapter, I present a summary of recent related work on performance, endurance and intrinsic in-memory processing capability of ReRAM crossbar arrays. A brief introduction to encoding techniques for NVM is also presented.

3.1 Performance of ReRAM Crossbars

3.1.1 Studies on RESET Operation

Since the RESET operation is one of the major performance bottlenecks for ReRAM crossbars, there have been many studies on reducing the RESET latency [105, 121, 114, 92, 119]. Xu *et al.* [105] proposed the double sided ground biasing (DSGB), multi-phase write operations, as well as a compression-based encoding approach to reduce RESET latency. Based on the observation that RESET latency correlates to the physical distance between selected row and and the write drivers, Zhang *et al.* [114] proposed to divide a crossbar array into several logical regions with different access latency, in order to exploit the discrepancy of RESET latency. Wang *et al.* [92] presented the write latency depends on worst-case data pattern in ReRAM crossbars, and proposed a voltage bias scheme to optimize write performance. Zhang *et al.* [119] proposed an ReRAM crossbar design with the double-sided write driver to reduce RESET latency. Additionally, a recent study [128] proposed several designs that are able to mitigate voltage drops and also shorten RESET latency for ReRAM crossbars.

3.1.2 Data Patterns in ReRAM Crossbars

Chang *et al.* [11] presented a similar observation for read operation to this dissertation. Mustafa *et al.* [67] and Shin *et al.* [83] reported that the detection margin for read operations depends on data pattern in ReRAM arrays. Deng *et al.* [18] discussed the worst-case data

patterns for read and write operations in an ReRAM crossbar array. Tang *et al.* [88] analyzed the impact of data pattern on the sensing current in ReRAM crossbars. Xu *et al.* [105] demonstrated that the RESET latency significantly increases as the number of reset bits (switched from "1" to "0") increases in ReRAM crossbars, and then exploited the data pattern to reduce RESET latency. Liang *et al.* [58] analyzed the voltage drop and data patterns in ReRAM crossbar arrays without selectors.

3.1.3 RESET Latency Discrepancy

Liang *et al.* [58] explored the correlation between data storage patterns and voltage drop in crossbar resistive memory without cell selectors. Zhang *et al.* [114] observed and leveraged the RESET latency discrepancy caused by row physical distance from write drivers to improve write performance. In this work, I preset, in addition to row address impact, the bitline data patterns also lead to RESET latency discrepancy in ReRAM crossbars.

3.2 Endurance of ReRAM Crossbars

3.2.1 Wear Leveling for Non-volatile Memories

Many prior work [74, 80] on enhancing PCM lifetime can apply to other resistive memories, and they shared the same general idea to evenly distribute write across all memory pages. Recent studies [123, 117] on wear leveling for non-volatile memories took process variation (PV) issue into consideration, which leads that different page has non-uniform endurance. However, compared to this work, they all ignored the impact of array structures on write endurance, and fail to exploit the intrinsic features in ReRAM crossbars.

3.2.2 Endurance for ReRAM Crossbar-based Neural Network Accelerators

Similar to crossbar ReRAM memory, the dot-product operation accelerators also suffer from limited write endurance when programming cells. Therefore, this work is critically im-

portant to crossbar resistive memory design as well as in-memory computing. Wear-leveling techniques for NVM based memories have been widely studied, which share a general idea of evenly distributing write accesses across pages [97, 123, 117, 74]. To address the endurance issue of ReRAM crossbar based neural network accelerators, a software and hardware co-optimization is proposed [116]. Unfortunately, this scheme only works for MLC ReRAM crossbars. Prior work [9] exploits gradient sparsification in neural networks and a row remapping scheme to improve ReRAM endurance, which is in fact complementary to my designs. A recent study on using low-precision weights [126] for CNN training can be also used to mitigate ReRAM crossbar endurance degradation. As shown in my evaluations, this is also orthogonal to the proposed design `ReNEW`, which is discussed in detail in Chapter 7, since it can further improve endurance in low bit-width weight matrices during the training.

3.2.3 Improving Endurance by Exploiting Stochastic Switching

An approximate switching scheme is proposed to improve the endurance in [6] for NVM based FF design, but it is lack of an analytical study between switching probability and enhanced lifetime.

3.3 Intrinsic In-Memory Processing Capability of ReRAM Crossbars

3.3.1 Current Accumulation Feature

The crossbar ReRAM architecture has recently attracted much attention [105, 99, 114, 82] owing to its smallest $4F^2$ planar cell size. In addition, due to its intrinsic analog current accumulation feature, the crossbar resistive memory is also adopted to accelerate dot-product operation based convolutional neural network computations [16, 85]. In the dissertation, I leverage this feature to profile and track the number of LRS cells along each bitline.

3.3.2 Neural Network Computing with ReRAM

Recent studies on neural network accelerators exploit the natural analog current accumulation feature of ReRAM crossbar architecture to implement dot-product calculations [16, 81, 85, 7, 24, 68, 21, 22, 53, 60, 71, 4, 36, 8, 26, 30, 9, 95, 73], wherein there are many [85, 9, 95, 73, 116] supporting neural networks training in ReRAM crossbars. A recent work [127] proposes to adopt SLC ReRAM crossbar to achieve reliable neural network computing, however, it does not exploit the better endurance of SLC ReRAM cells for neural network training.

3.4 Data Encoding for NVM

DCW [109] and *Flip-n-Write* [17] were proposed to reduce the amount of NVM cells to be modified during programming time, by which the lifetime of NVM cells can be improved. The data encoding scheme from [105] was proposed to reduce the number of RESET operations. However, *Flip-n-Store* proposed in this dissertation, which is discussed in detail in Chapter 8, has a different motivation from all prior work, and it aims at limiting the number of LRS cells in 3D-VRAM arrays at runtime.

4.0 Observations

4.1 The Correlation between RESET Latency and the Number of LRS Cells

The relationship between cell RESET switching time and IR drop on the target cell can be modeled using Equation 4.1, as shown in recent studies [105, 27].

$$t \times e^{kV_d} = C \tag{4.1}$$

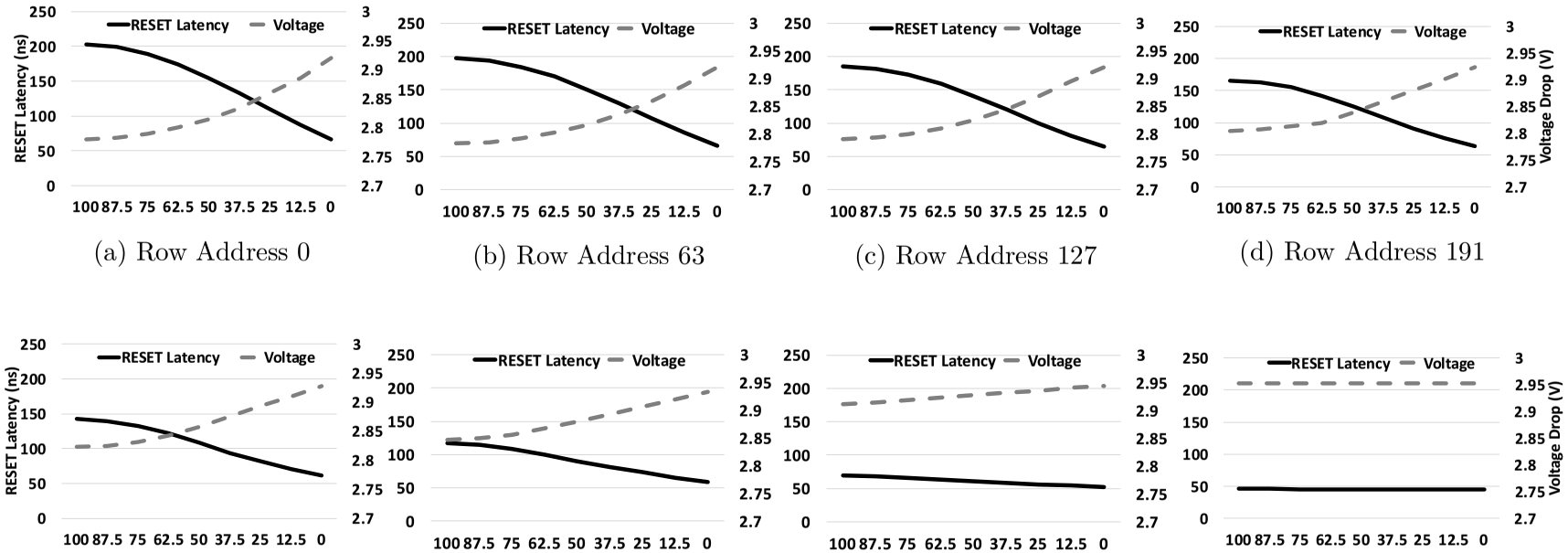
where `t` denotes cell RESET switching time; V_d denotes the voltage drop across the targeted cell; `C` and `k` are experimental fittings constants extracted from prior studies. From the equation, the cell switching time is highly sensitive, i.e., exponentially inverse correlation, to the voltage drop. A voltage drop of 0.4V results in 10× RESET latency increase [27].

During RESET operation, half-selected cells do not change state and exhibit as resistive devices. Given the same voltage stress, a half-selected cell in LRS would have larger sneak current than the one in HRS. Similar observation was reported for read operation in [82].

Given one selected wordline and one selected bitline, the correlation among IR drop, the number of LRS cells, and RESET latency is studied. Figure 6 summarizes the correlation for rows with different row addresses — `Row 0` and `Row 511` are the farthest and the closest rows to the write driver, respectively. The y-axis shows the RESET latency (left) and IR drop (right) while the x-axis shows the percentage of LRS cells in the selected bitline[1]. I focus on bitline LRS cells and assume the worse case for the wordline in this work. The impact from wordline tends to be smaller due to the adoption of DSGB [105] and each subarray saving 8 bits from one cacheline. I study the RESET latency in this work, a similar observation for READ was reported in [11]. In the experiments, the Verilog-A model from [40] is adopted to build and simulate a 512 × 512 Mat circuit model in HSPICE. Table 2 summarizes the ReRAM crossbar model parameters.

From the figure, given a row, e.g. `row 0`, the more LRS cells there are in the bitline, the larger IR drop the sneak current brings, and the longer time the RESET operation takes.

[1]Note that the term of in-memory data patterns used in this work refers to the percentage of LRS cells along bitlines, i.e., it is to characterize low architectural level data layout, similar to that in prior work [11, 58].



(a) Row Address 0 (b) Row Address 63 (c) Row Address 127 (d) Row Address 191

(e) Row Address 255 (f) Row Address 319 (g) Row Address 447 (h) Row Address 511

Figure 6: Subfigures (a) to (h) show that the variations of RESET latency and voltage drop at different LRS cell percentages in bitlines when accessing to different row address in ReRAM array. The Row Address 0 is the farthest row from driver, and Row Address 511 is the nearest row to the driver.

Another observation is, the impact diminishes as the row becomes closer to the write driver. For `row 511`, the RESET latency is small and indistinguishable for the cases with different percentages of LRS cells.

Table 2: ReRAM model parameters

Metric	Description	Value
A	Mat Size: A wordlines × A bitlines	512×512
n	Number of bits to read/write	8
Iw	Cell current at Vw	$88\mu A$
Rwire	Wire resistance between adjacent cells	2.82Ω
Kr	Nonlinearity of the selector	200
Vw	Full selected voltage during write	3.0V
Vread	Read voltage	1.5V
-	Voltage biasing Scheme	DSGB

Prior studies [11] have revealed that, with a larger percentage of LRS cells on bitlines, the bitline discharging time (developing time) increases during the read operation. However, ReRAM read and SET operations are much faster than ReRAM RESET operations — ReRAM read and SET are 18ns and 10ns, respectively, while RESET ranges from 56.4ns to 202.4ns. In this dissertation, I focus on optimizing ReRAM RESET operations. While the proposed schemes are applicable to optimizing read and SET operations, further study is necessary to evaluate the tradeoff between limited performance improvement and increased hardware complexity.

4.2 Endurance Variation in ReRAM Crossbars

A recent study [115] revealed a tradeoff between write latency and endurance of ReRAM cell — the endurance degrades when write latency increases. The relationship can be analytically modeled using the following equation:

$$Endurance \approx (\frac{t_W}{t_0})^C \tag{4.2}$$

where t_W is write latency, t_0 and C are constants. In this work, I choose the same $C = 2$ as in [115] to model a quadratic correlation between write endurance and latency.

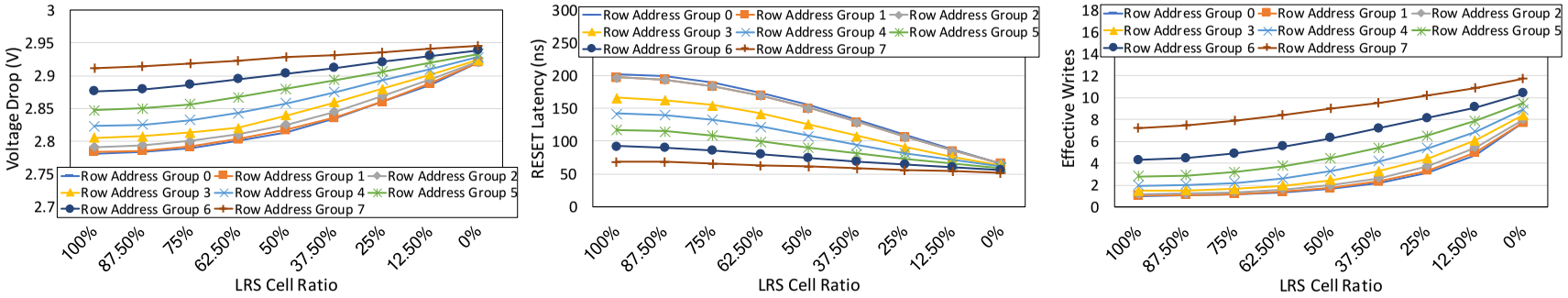
As that IR drop results in RESET latency discrepancy among the ReRAM cells due to different physical locations and dynamic bitline data patterns. According to Equation 4.2, the cells in ReRAM crossbar would exhibit endurance discrepancy. Figure 7 summarizes the endurance discrepancy across the crossbar. I divide 512 rows to eight address groups with each group containing consecutive 64 rows. *Row Address Group 0* is the one that is the closest group to the write drivers. *LRS cell ratio* indicates the percentage of LRS cells in one bitline. I adopt the worst-case voltage drop and RESET latency in every 64 rows to represent one *Row Address Group*.

From Figure 7a and 7b, the more LRS cells on selected bitlines, the larger sneak current flows through half-selected cells. Thus smaller voltage drop and longer RESET latency are observed. Also, the farthest rows from write drivers are more vulnerable to the impact of bitline data patterns on RESET latency. The observation is similar to that in [99, 114]. In conclusion, the discrepancy of RESET latency leads to write endurance variation in ReRAM crossbar.

4.2.1 Effective Write

In this work, I use **effective write** to summarize the overall wearing effect of one write at runtime. Intuitively, let us assume that one cell can sustain 10^5 times writes if using write pulse width X and 10^6 times writes if using write pulse width Y. Assume other conditions are the same. It is concluded that each write with pulse X corresponds to ten writes with pulse Y. According to Equation 1, the effective write depends on the write pulse width while an optimized write strategy [99] chooses pulse width based on (1) target row address and (2) the numbers of LRS cells in the bitline. Therefore, the actual effective write depends on the latter two factors.

Figure 7c depicts the relationship between effective writes and row addresses and LRS ratios. In my experiments, when writing *Row Address Group 0* with 100% LRS cell ratio, the write takes longest duration to complete. Such a write has the smallest wearing effect, as



(a) (b) (c)

Figure 7: Subfigures show that the variations of (a) voltage drop on selected cells and (b) RESET latency and (c) effective writes at different LRS cell percentages in bitlines when accessing to different row address in ReRAM array. The Row Address Group 0 represents farthest rows from drivers, and Row Address Group 7 consists of nearest rows to the drivers.

shown in Equation 1. All other writes are normalized to this baseline, that is, the effective write of writing address group 0 under 100% LRS cell ratio is the normalized '1'. For all other writes, the effective writes are calculated with following equation:

$$EW = \left\lceil (\frac{t_L}{t})^2 \right\rceil \tag{4.3}$$

where t_L is the longest write latency (i.e., writing group 0 with 100% LRS ratio); and t is the actual write latency of the given write.

4.2.2 Design Challenge

Given that writes to ReRAM crossbar exhibit different effective writes at runtime, to extend chip lifetime, effective writes across all ReRAM cells should be evenly distributed. Unfortunately, existing wear leveling approaches evenly distribute raw writes across all ReRAM cells. As a result, it is highly possible that rows in the address group 7 are worn out while the rows in the address group 0 are very healthy.

There are two families of wear leveling schemes: one is to track writes to blocks using a table and periodically mitigate the block that is stressed the most [117, 123, 19]; the other is having physical addresses randomly mapped to device addresses and periodically changes to a new random mapping [74, 80]. In this dissertation, I propose a table based wear leveling scheme that evenly distributes effective writes at runtime, and leave the development of randomized mapping based wear leveling on effective writes as the future work.

5.0 Speeding Up RESET Operation

5.1 Low-Overhead Runtime Profiling

In this section, I present an overview of my scheme, elaborate the details of the low-overhead runtime profiler and then propose the compression based optimization for further performance improvement. Finally, I illustrate the profiling scheme with an example and estimate the overhead.

5.1.1 An Overview

Figure 8 presents an overview of the proposed scheme. Each cacheline is assumed to have 64B or 512 bits. These bits are saved in 64 mats spreading across 8 chips and each mat saves 8 bits from the cacheline, the same as previous work [105]. The 8 corresponding bitlines saving these 8 bits form a group. Two cachelines are mapped to use the same 8-bitline group, e..g. `a0` and `a1` use the first group, if their device addresses are separated by `K`, here `K` is a multiple of 64 depending on the number of mats, and line address interleaving. The cachelines that share the first 8-bitline group are `a0+i×K (0≤i<512)`, which are referred to as the *bitline-sharing-set* in the following discussion.

Worst-case bitline flag. A 3-bit flag `W-Flag` is attached to each bitline-sharing-set. The flag records the worst case bitline of all 512 bitlines shared by this set. In practice, the worst case bitline of each 8-bitline group in one mat is first found, and then the worst case from 64 mats is found. Since one mat has 512 rows, the number of LRS cells on one bitline varies from 0 to 512. Instead of recording the accurate number, the range [0..511] is divided into 8 subranges such that a 3-bit flag `W-Flag` can denote its subrange, e.g., '000' denotes subrange [0..63] and '010' denotes subrange [128..191].

In the next section, I exploit a runtime profiler that periodically detects the worst case bitline in each mat as well as the worst case for the whole *bitline-sharing-set*.

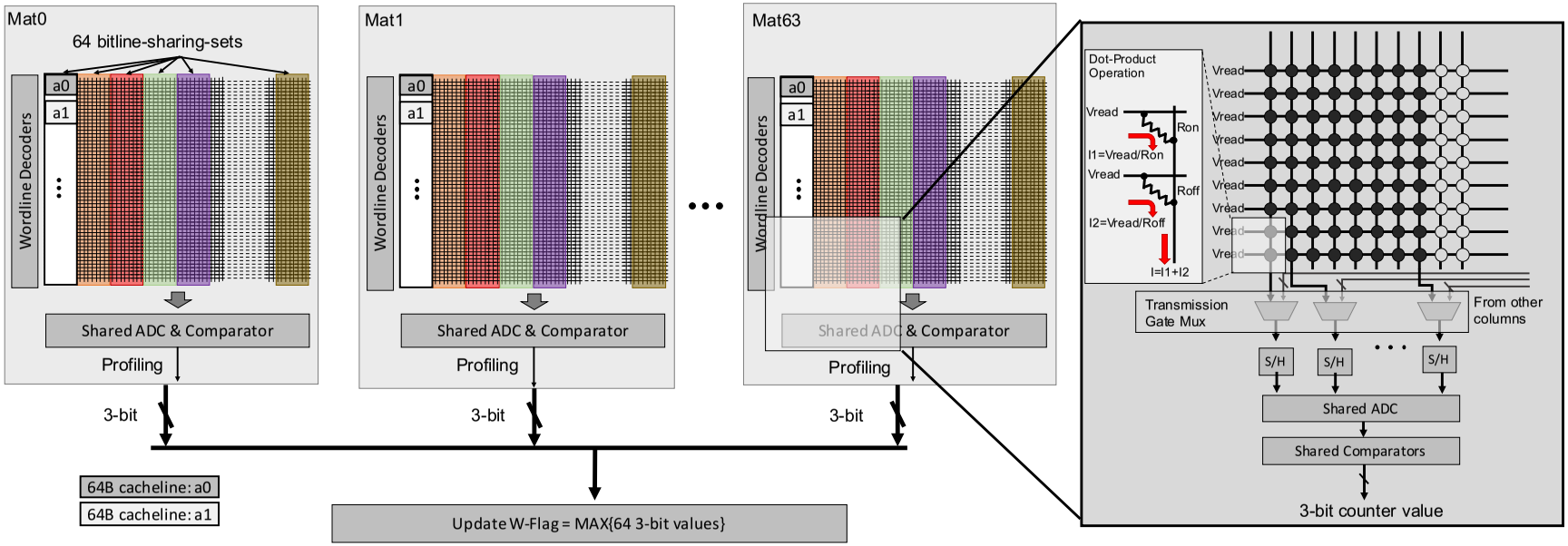


Figure 8: An overview of the proposed low-overhead runtime profiling scheme.

Tracking the worst-case. A 6-bit counter `W-Cnt` is attached to each *bitline-sharing-set*. The counter is cleared each time when the worst-case flag is updated, that is, either after profiling update or due to `W-Cnt` overflow (as follows).

At runtime, the counter is incremented for each memory write that falls in the *bitline-sharing-set*. This is based on the most conservative assumption that the write always introduce one more LRS cell on the worst-case bitline among all 512 bitlines shared by *bitline-sharing-set*. A counter overflow event increments `W-Flag` if `W-Flag` does not saturates. The counter is then cleared. I will elaborate the use of `W-Flag` and `W-Cnt` in following sections.

RESET latency optimization. To RESET a memory line, its `W-Flag` and physical address are fetched to determine the appropriate tWR time for the RESET operation. By looking up a pre-tested RESET latency table stored in memory controller, always using the most conservative timing for each write can be avoided. For example, if `row 0`'s `W-Flag` is '010', a tWR timing of 154.6ns may be used instead of 202.4ns in the baseline design. The quantitative values of tWR timing come from the HSPICE circuit simulations, which will be discussed in a later section.

I next elaborate the design details and illustrate the overall workflow with examples.

5.1.2 Design Details of Runtime Profiling

I first describe the runtime profiling mechanism that faithfully tracks the number of LRS cells in each bitline. Clearly, reading all memory lines from the mat for detection would introduce prohibitive overhead. In this work, I leverage the current aggregation feature of ReRAM crossbar array [32], which has been widely exploited for accelerating in-memory computation [16, 81, 85, 7]. Most existing memory profiling technique are for offline test. For example, *march test* [90] was proposed for checking memory data integrity. The test cannot be adopted at runtime as it can be as slow as 0.4ms per row [118, 76, 90], which is much longer than regular ReRAM read or write operation latency.

Figure 8 illustrates how the proposed profiling scheme works. When there is a need to profile, the memory controller sends out a profiling command with a 18-bit digial ID number (which is enough to guarantee a unique ID for each *bitline-sharing-set* in a 8GB memory

system) for determining the *bitline-sharing-set* in 64 mats. For each mat, all (512) wordlines and the eight bitlines that belong to the *bitline-sharing-set* are activated for performing profiling operation. This is similar to dot-product operation in [16].

As shown in the figure, all wordlines are applied with V_{read}; the selected eight bitlines are applied with 0V; and all other bitlines are applied with V_{read} to depress sneaky currents. The currents flow through the eight bitlines are highly correlated to the number of LRS cells. The more LRS cells, the larger current will be applied to ADC and comparator circuits that are shared by all 64 8-bit read/write groups. I adopt the analog to digital conversion circuitry developed for accelerating in-memory computation. The bitline profiling currents are first sent to analog transmission muxes, which select the appropriate *bitline-sharing-set* to profile. The currents are then fed to sample-and-hold (S/H) logic and the ADC unit. After the analog to digital conversion, the largest current (corresponds to the worst-case bitline in this mat) is represented as a 3-bit digital value.

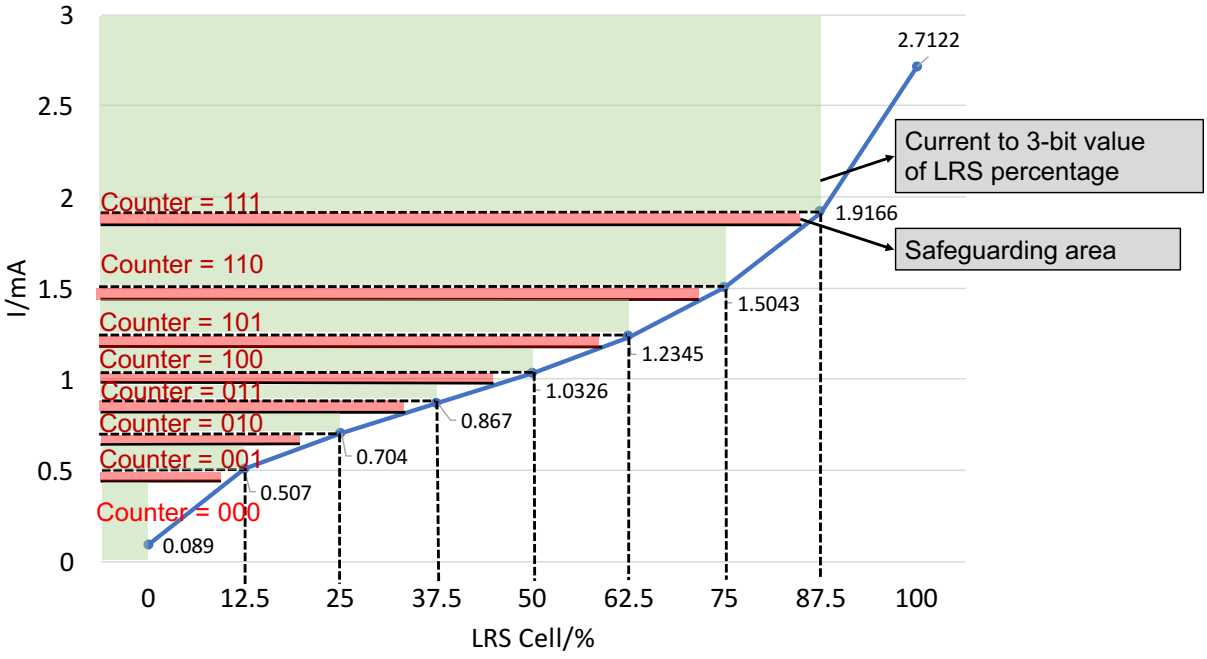


Figure 9: The profiling current vs. LRS cell percentage in 512×512 ReRAM crossbar array.

I divide the range [0..511] into eight subranges with equal size (except the last one which has one more value). As shown in Figure 9, the mapping from bitline currents to subranges is

set up before profiling. To account for runtime voltage fluctuation and cell process variations, a $0.1mA$ guard band is allocated for each subrange. That is, subrange '011' corresponds to LRS cell percentage range [37.5%..50%), the bitline profiling current is $1.03mA$ if there are 255 LRS cells in one bitline. For high reliability, a bitline is tagged as '011' as if the profiling current is $0.93mA$, that is, a line may be tagged to have more LRS cells than it actually has.

The `W-Cnt` tracks the write to the *bitline-sharing-set* after profiling. By default, the memory controller profiles the set again after 64 writes so that 6-bit value is used to represent `W-Cnt`. When `W-Cnt` overflows, it is possible to either re-profile the bitlines or increment `W-Flag` directly (before it overflows). Given ReRAM writes not always introduce more LRS cells to the worst-case bitline, it is beneficial to periodically profile the set.

5.1.3 Determine the RESET Timing

At runtime, the physical address and `W-Flag` are used to determine the appropriate tWR timing for the RESET operation. The reason that I also use the row address is that, similar as that in [114], row RESET latency also depends on its row index in one mat, i.e., the distance to the write drivers — given the same percentage of LRS cells along the bitlines, `row 0` and `511` have the largest and smallest RESET latencies, respectively. Therefore, the 512 rows in one mat are split to eight address subranges, and the worst case RESET of this subgroup is used to write cachelines in each range, as shown in Figure 10.

Table 3 summarizes the write timing (tWR) of RESET operation with different LRS cells along bitlines and different row address category. The table is kept in the memory controller, which is used in scheduling write operations to ReRAM memory. The quantitative values of RESET operation timing are from my simulations of a 512×512 Mat circuit model in HSPICE with parameters shown in Table 2.

An example. I next use the example in Figure 11 to illustrate how my proposed online profiling works and how to determine the timing of RESET operations based on `W-Flag` and `W-Cnt`.

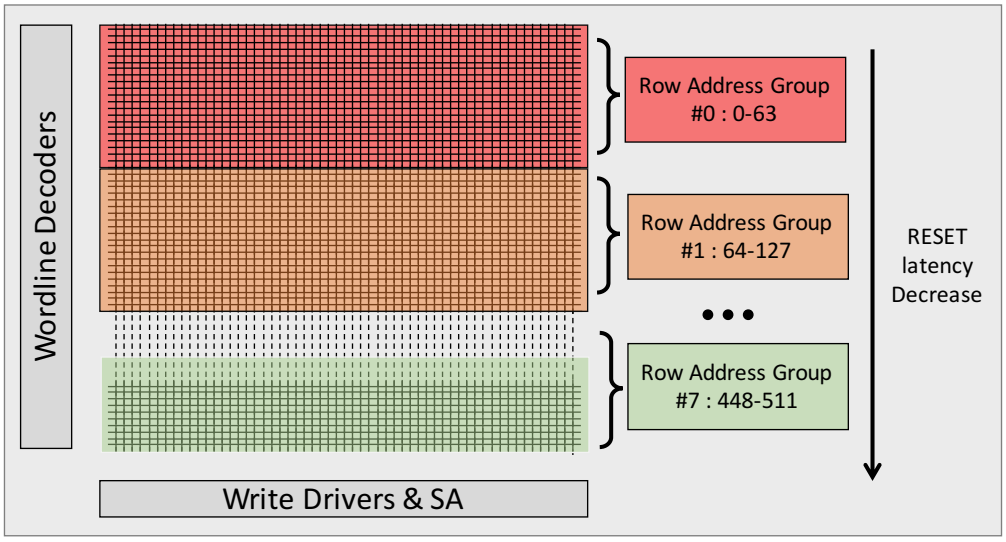


Figure 10: The rows with different addresses are mapped to 8 groups with different worst-case RESET latencies.

Table 3: The tWR (ns) for RESET operation

LRS	Row Address Group							
Ratio	0	1	2	3	4	5	6	7
111	202.4	197.7	184.9	165.9	142.3	117.2	92.4	69.1
110	202.4	197.7	184.9	165.9	142.3	117.2	92.4	69.1
101	199	194	181.8	162.9	139.8	115	90.5	68
100	189	184.3	172.6	154.8	132.9	109	85.8	65.5
011	173.8	169.7	158.5	142	121.9	99.8	80.2	63.4
010	154.6	150.9	140.9	126	107.9	90.3	74.7	60.9
001	132.9	129.3	120.9	107.9	93.9	81.3	69.2	58.8
000	109.7	106.9	99.7	90.8	81.8	73.2	64.5	56.4

Table 4: Comparing the profiling overhead in one bank

Comp.	Params	Spec	Power/Energy	Area (mm^2)
ADC [49]	sampling speed resolution number	1.28GS/s 8-bit 8	24.48mW	0.012
S+H [81]	number	8×64	5uW	0.00002
ReRAM array	Mat number Mat size	1024 512×512	Regular Profiling: 267.178pJ Fine-grained Profiling: 168.332pJ Read: 72.842pJ Leakage: 255.233mW	2.078

5.1.3.1 Online Profiling Operation A profiling operation is always triggered by a `W-Cnt` overflow. The default profiling frequency is after every 64 writes to the same 6-bit `W-Cnt` flag). For the example in Figure 11, `W-Cnt` of *bitline-sharing-set* with an ID `0x004ff` overflows, which sends a profiling command to all 64 corresponding mats (❶), each of which contains 8 bitlines. It then performs the dot-product fashion profiling within each mat (❷) and produces a 3-bit counter that maps the aggregated bitline current to a LRS cell subrange. Each subrange indicates the worst-case LRS cell percentage of the corresponding mat (❸). `W-Flag` of *bitline-sharing-set* `0x004ff` is then updated with the maximum (the very worst-case) of all 64 subrange values (❹). At last, `W-Cnt` is reset to zero, which completes one online profiling operation.

5.1.3.2 Write Operation with Optimal RESET Timing With the proposed profiling scheme, the timing of RESET operations is determined by looking up an optimal RESET timing table at runtime. For the example in Figure 11, a RESET operation to logic cache-

line `a7` is being served. Based on its physical address, the row address group number (❺) and *bitline-sharing-set* ID (❻) (`0x004cd` in this case) are identified first, and an up-to-date `W-Flag` (❼) is fetched. Then the optimal RESET timing is found in Table 3 (❽) and `W-Cnt` is incremented. For the cells that need to be RESET and fall in *bitline-sharing-set* `0x004cd` across 64 mats (`a7<0:7>` ... `a7<224:231>` ... `a7<504:511>`), the RESET operations can finish within the optimal RESET timing (❾).

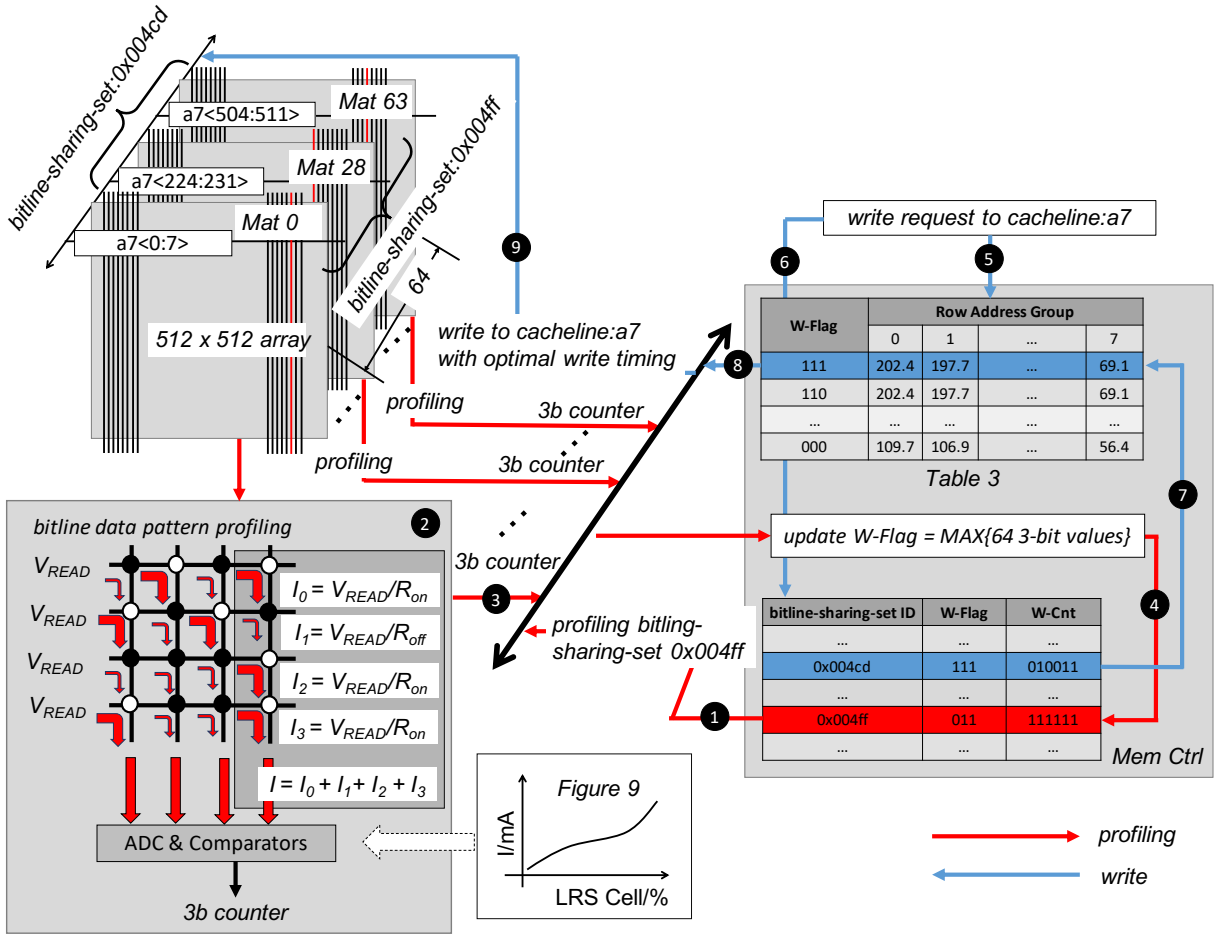


Figure 11: An example of how my proposed online profiling works and how to determine the RESET timing.

5.1.4 Reduce Bitline LRS Cells

Based on the observation that RESET latency depends on the number of LRS cells along bitlines, it is important to reduce the number of LRS cells in the crossbar. A simple optimization is to save the cacheline in compressed format [2] and fill in unused cells with 0s, i.e., RESET them to HRS. However, I observed a direct application of data compression exhibits little help — the RESET latency is hardly changed. This is because the RESET latency depends on the worst case of all 512 bitlines. Assume every cacheline in a *bitline-sharing-set* can be compressed to its half size and thus uses 256 cells. If every cacheline uses the first 256 bitlines, there would be zero LRS in the other 256 bitlines. Unfortunately, it is of little help because the worst case bitline may stay in the first 256 bitlines.

I therefore propose a row-address biased data layout to distribute extra 0s evenly to all bitlines. Given one *bitline-sharing-set* `a0+i`$\times$`K (0`$\leq$`i<512)` where `a0` is the cacheline address that is mapped to the first row. When saving a compressed cacheline in, e.g., `row i`, the row starting address is shifted to the right by `i` bits and then the unused cells are filled in with 0s in the row, as shown in Figure 12.

5.1.5 Overhead Analysis

Profiling overhead. The overhead comes mainly from runtime profiling. After every 64 writes to one *bitline-sharing-set*, the memory controller sends out one profiling command, which activates 64 mats. In each mat, all rows and eight bitlines are activated.

Table 4 summarizes the overheads for each ReRAM memory bank. I evaluated the power consumption and area by HSPICE simulation and NVSim [20] at 32nm. A profiling operation consumes about 3.7x read energy. For either read or profiling, a huge portion of the power is consumed by internal I/O and row/column decoders, thus the energy consumption is not linear to the number of opened rows.

I followed recent studies [81, 49] to estimate the power and area overheads of adopting ADC and sampling and holding circuits. I used eight ADC units in each bank. An ADC has 1.28GS/s sampling speed and introduces 50ns profiling latency. In the experimental section

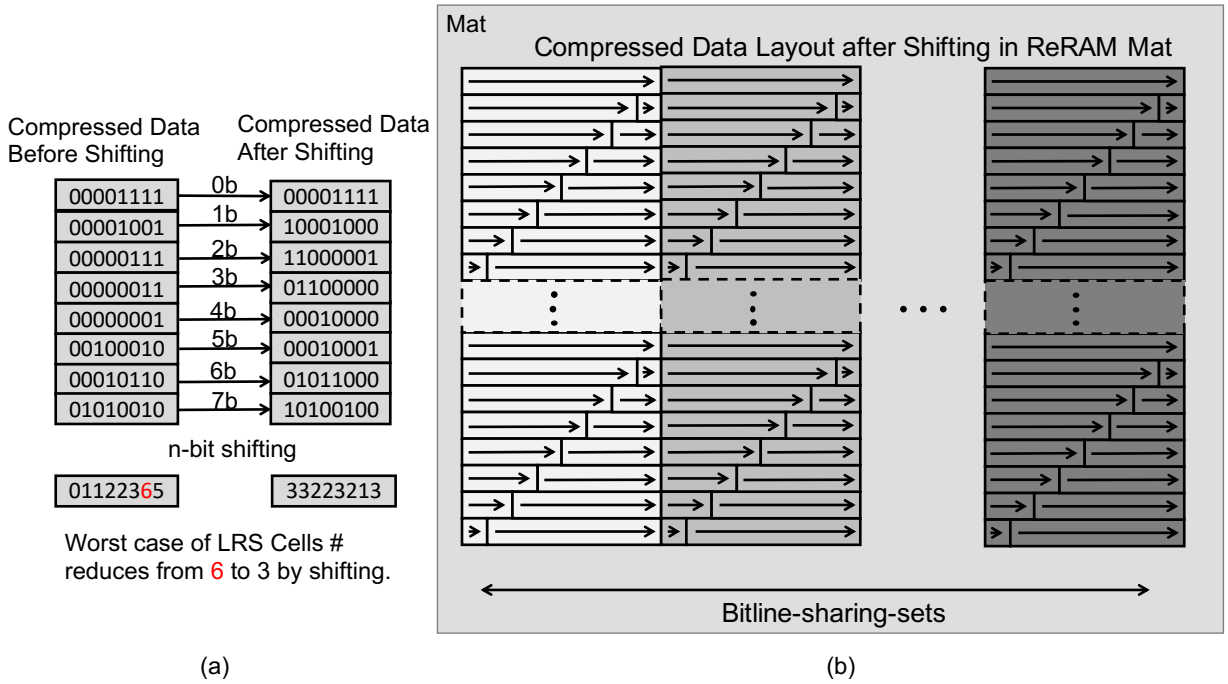


Figure 12: Reducing LRS cells through data compress: (a) logic view; (b) shift in each mat.

of this chapter, I will study the performance and power efficiency with different numbers of ADC units.

A profiling command return 3 bits from each activated mat. As a comparison, a read operation returns 8 bits from each mat. Therefore, the profiling results are returned to the memory controller through data bus, without introducing additional overhead other than a regular read.

Counters storage and RESET adjustment. One 3-bit `W-Flag` and 6-bit `W-Cnt` are attached to each *bitline-sharing-set*. A *bitline-sharing-set* contains 512 64B memory lines, or 32KB data. For a 8GB memory system, about 288KB storage is needed to hold all flags. In this work, I keep all flags in the memory controller for simplicity. In the future work, I will keep a small buffer hold a subset of flag while keeping the rest in the L2 cache. The RESET operation can be issued in parallel to the table lookup. Due to long RESET latency, the table lookup result can be returned at a later time to the memory controller to determine when to terminate RESET operation. I expect negligible performance overhead.

5.2 Profiling Optimization

Even though online profiling helps to optimize RESET latency and thus improve write performance, it introduces non-negligible profiling overhead, including performance overhead and energy consumption overhead. While the former is small as I shall show in the experiments, the latter is much larger due to the large energy consumption from ADC units. I focus on optimizing profiling energy consumption in this section.

5.2.1 Profiling Energy Overhead Analysis

To better illustrate the profiling energy overhead, the dynamic energy dissipation of ReRAM memory on read, write and profiling operations are compared, respectively, for a

wide range of benchmarks[1], and summarize the results in Figure 13. From the figure, I observe that the profiling energy consumes an average of 13.4% of total dynamic energy, a non-trivial portion of memory energy dissipation. Thus, it is important to optimize online profiling to reduce the profiling energy overhead.

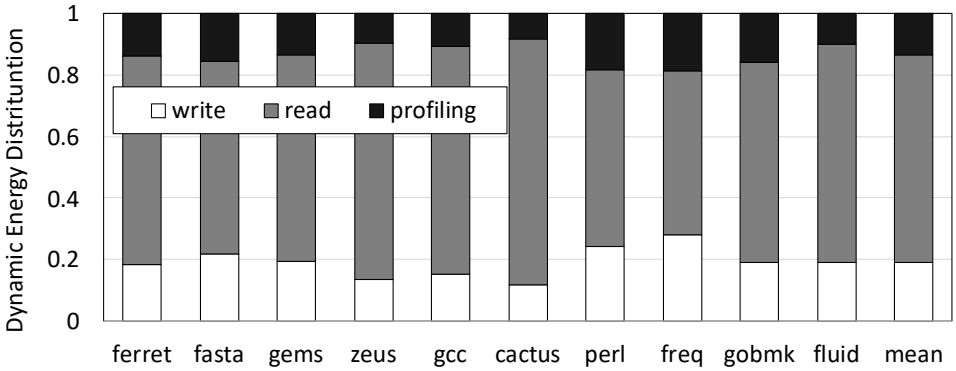


Figure 13: The dynamic energy distribution when adopting the proposed profiling technique.

I next propose two optimization schemes to mitigate the overhead by reducing the number of cells to be activated at profiling.

5.2.2 Selective Profiling

Figure 14 presents the basic idea of selective profiling. When performing the N-*th* round profiling for a *bitline-sharing-set* at runtime, I find out that the worst-case LRS-cell-per-bitline number is 384 out of 512 cells, as shown by the red bar in Figure 14a. However, it occurs only in one mat while the worst-case numbers from other mats are much smaller. In the figure, the green bars represent the numbers that are smaller than 256. For the mats corresponding to the green bars, the worst scenario during the next profiling interval occurs when every write operation increments the number of LRS cells in those mats. Given the default profiling frequency is every 64 writes, the worst scenario may introduce at most 64 more LRS cells, i.e., the worst LRS-cell-per-bitline numbers for these mats would still be smaller than 384 by the end of the next profiling interval. Since the red bar is already 384

[1]The experiment and simulation methodologies are discussed in Section 5.3 in detail.

at the beginning of the next profiling interval, it is safe to assume the worst case for the green bar mats and skip profiling them in the next profiling interval. However, for the mats corresponding to the red bar and the gray bars in the figure, the N+1-*th* round profiling still needs to be performed.

To implement the proposed selective profiling scheme, I group every two consecutive profiling rounds together and make the i-th round profiling a regular profiling (i.e., the same as that in the baseline profiling) while the (i+1)-th round profiling a selective profiling (i.e., it is applied only to a subset of mats). The regular profiling and selective profiling rounds are performed alternately. In particular, after collecting the 3-bit flags from all 64 mats during a regular profiling, the memory controller constructs a 64-bit `profiling mask` with each bit representing whether the corresponding mat needs to perform selective profiling for the next round. The bits are initialized as `1s` and updated based on the difference between its 3-bit flag and `W-Flag`, the worst-case of all mats. Assume the 3-bit flag from mat j is $\texttt{W-Flag}_j$. If $\texttt{W-Flag}_j+2 \leq \texttt{W-Flag}$, i.e., the worst LRS-cell-per-bitline number from one mat is at least 128 smaller than the worst LRS-cell-per-bitline number of all mats, the corresponding bit of the mat in the `profiling mask` is set to `0`; otherwise, the `profiling mask` bit is kept as `1`. For the next selective profiling round, the mats whose `profiling mask` bits are 0s are not profiled.

Given selective profiling only skips the profiling operations on a subset of mats, it does not degrade write performance and reliability. Its benefits come from two folds: 1) it helps to save the energy consumption on the ADC/S+H circuits and the multi-row read operations on ReRAM arrays; 2) it shortens the ADC latency at the sampling stage. This is because fewer samples from mats need to be processed for analog-to-digital conversion. In Section 4.4, I study the performance and energy efficiency improvements in my experiments.

5.2.3 Fine-grained Profiling

I next propose to reduce the profiling overhead as shown in Figure 15. As aforementioned, the $V_{\texttt{READ}}$ voltage is applied to all wordlines in order to profile the ratio of LRS cells along the bitlines within the *bitline-sharing-set*. These simultaneous read operations contribute to

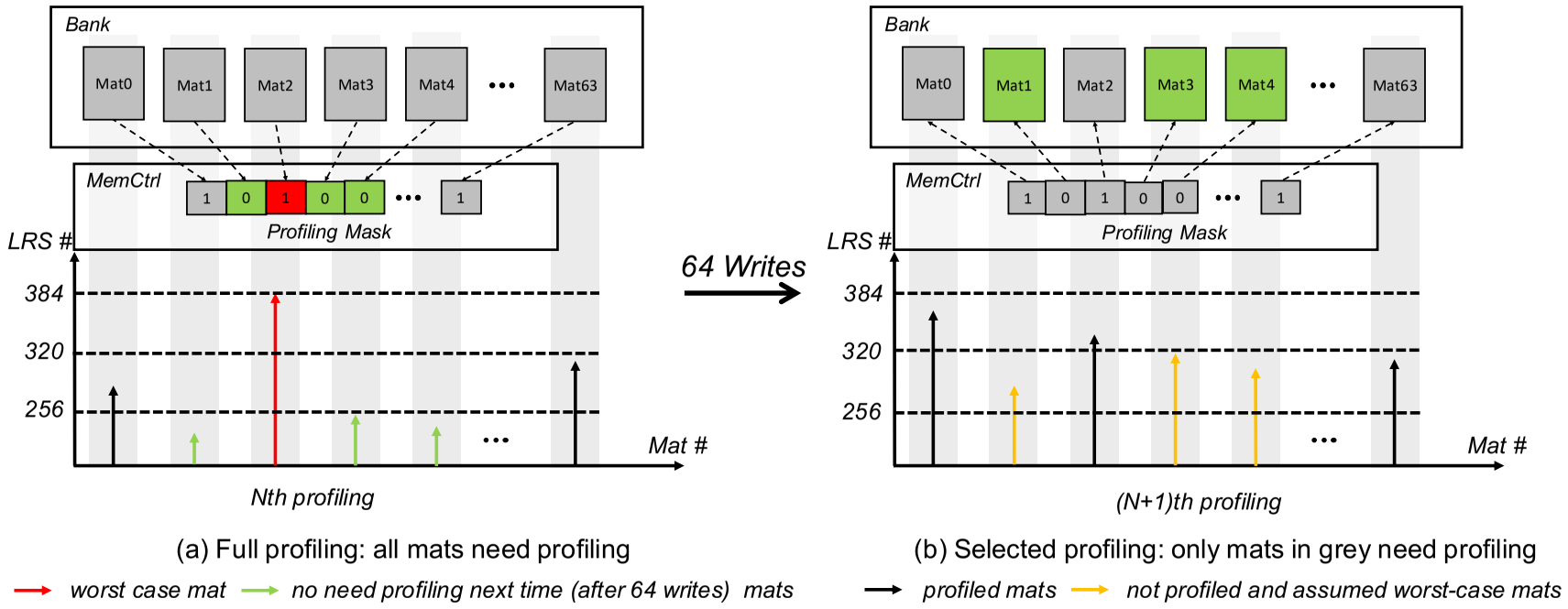


Figure 14: The scheme of proposed selective profiling.

the active energy consumption of profiling overhead. Intuitively, by reducing the number of wordlines that are opened to read, the profiling overhead can be mitigated.

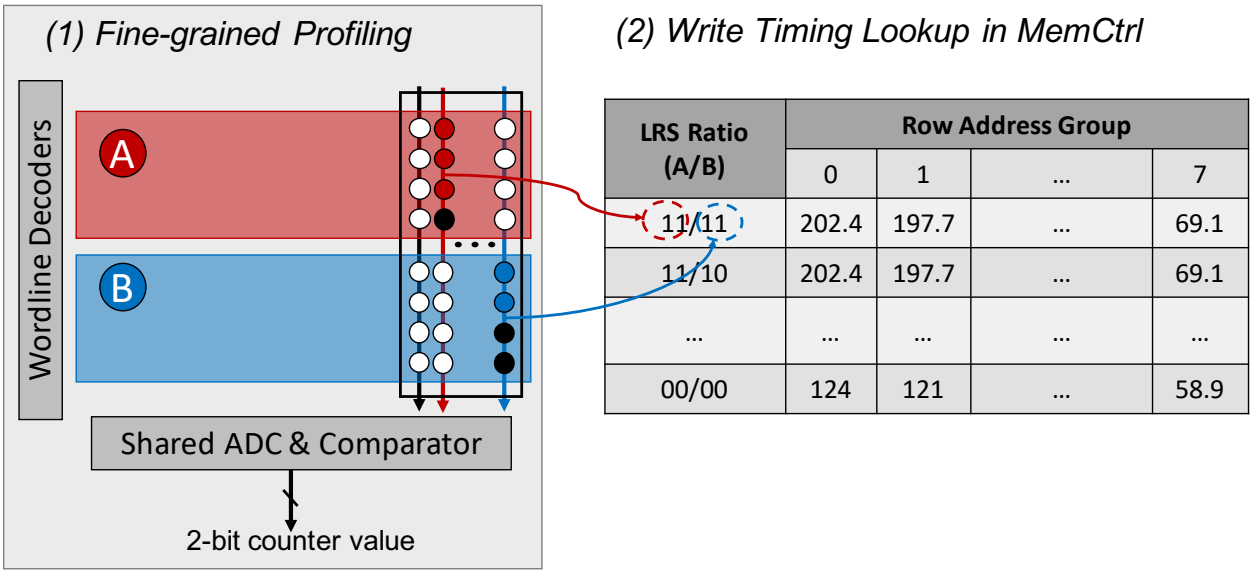


Figure 15: The scheme of proposed fine-grained profiling.

Based on this observation, one 512×512 ReRAM mat is split into two 256×512 sub-mats. In Figure 15, they are labelled as "A" and "B", respectively. Each sub-mat consists of 4 row address groups. Each sub-mat is profiled independently and two sets of `W-Flag` (2-bit `W-Flag-A` and `W-Flag-B`) and `W-Cnt` (6-bit `W-Cnt-A` and `W-Cnt-B`) counters are used to track the profiling results and to determine the RESET timing. By keeping the same profiling frequency, i.e., each sub-mat needs to be re-profiled after accumulating 64 writes, the same total number of profiling operations are kept. The profiling procedure, including detecting runtime bitline data patterns and tracking the worst-case flag within one *bitline-sharing-set*, is similar to the baseline profiling. The only difference is that the bitline data patterns for each sub-mat are profiled separately. For the profiling, a 2-bit value is enough to denote `W-Flag-A` and `W-Flag-B` the with the same accuracy as the baseline profiling as the number bitline LRS cells ranges from 0 to 256 in each sub-mat.

Determining the RESET timing is slightly different in the fine-grained profiling design. As shown in Figure 15, the two LRS ratio numbers (from sub-mats A and B, respectively) need to be combined to determine the optimal timing. Since conservative estimation is

adopted, the combination may lead to over-estimation, which slightly degrades the choice of the optimal timing.

Comparing to the baseline profiling, the fine-grained profiling scheme exhibits many advantages: (1) It activates a smaller number of wordlines and thus reduces the dynamic energy consumption. My study shows that, when activating 256 wordlines during profiling, the fine-grained profiling consumes 63% energy of the one that activates all 512 wordlines (Table 4). (2) Instead of having 3-bit `W-Flag` values transferred across the memory interface, 2-bit `W-Flag-A` and `W-Flag-B` values are returned, which may potentially save the memory bandwidth. (3) The fine-grained profiling potentially enables the finer tuning of RESET latencies.

5.3 Experimental Setup

In this section, I present the modeling and simulation methodologies for evaluating the energy and performance of ReRAM crossbars.

5.3.1 Modeling and Simulation Methodologies

To evaluate the effectiveness of my proposed design, in addition to the HSPICE modeling and simulation as introduced in Chapter 4, I used an in-house simulator to simulate the proposed ReRAM access scheme and compare it to the conventional and state-of-the-art designs. Table 5 summarizes the configuration for the baseline system. I plugged the numbers from HSPICE and NVSim [20] simulations into my architectural simulator to obtain the performance and memory energy efficiency results. I used Pintool to generate memory access traces from SPEC2006 [31], PARSEC [5] and BioBench [3] benchmark suites.

5.3.2 Workload Characterization

Table 6 characterizes all benchmarks used in the experiments. I carefully chose a subset of benchmarks with different memory access WPKI and RPKI in order to study the effectiveness

of my design. The benchmarks are categorized to three types: High, Medium and Low, respectively, according to their memory access intensity.

5.3.3 Schemes for Evaluations

In this work, I implemented and compared five different schemes, including the conventional and state-of-the-art ReRAM designs as follows:

- `BL` — This scheme is conventional ReRAM crossbar design. The baseline adopts DSGB voltage driver for latency reduction.
- `RA` — This scheme is the state-of-the-art design [114] that adopts row address awareness technique to reduce RESET latency.
- `LRS` — This scheme is the naive design that only adopts data pattern profiling technique.
- `CMP` — This scheme is built on top of `LRS`. It adopts data compression and shifts the rows starting bits based on its row addressed within each mat.
- `PROF` — This scheme is built on top of `CMP` and includes all enhancements in the work. In particular, it adopts a two dimensional tWR timing table (as shown in Table 3) in determining RESET latency.

I also evaluated the effectiveness of following three schemes with profiling optimization techniques:

- `SEL_PROF` — This scheme is built on top of `PROF` and adopts the selective profiling scheme to save energy.
- `FINE_PROF` — This scheme is built on top of `PROF` and adopts the fine-grained profiling scheme.
- `SEL_FINE_PROF` — This scheme adopts both profiling optimizations to mitigate profiling overhead.

In system performance evaluation, the proposed profiling techniques is also compared with `IDEAL_PROF`, the scheme that assumes zero performance overhead.

Table 5: System configuration

Processor	4 cores; single issue in-order CMP; 4GHz
L1 I/D-cache	Private; 16KB per core; 4-way; 2 cycle latency
L2 cache	Private; 1MB per core; 8-way; 64-byte block size; 10 cycle latency
Main memory	8GB; 1 channel; 2 ranks; 8 chips/rank, 2Gb x8 ReRAM Chip, 8 banks/chip; 1024 mats/bank; scheduling reads first, issuing writes when there is no read, issuing write burst when W queue is full
ReRAM Timing	Read Latency 18ns@1.5V; SET latency 10ns@3V; RESET latency refers to Table 3@-3V, 88μA

Table 6: Benchmarks characterization

Memory Intensity	Name	Benchmark Suite	WPKI	RPKI
High	ferret	PARSEC	12.44	19.44
	fasta_dna	BioBench	9.36	11.88
	gemsfdtd	SPEC2006	6.27	9.82
	zeusmp	SPEC2006	1.62	4.12
Medium	gcc	SPEC2006	1.44	3.21
	cactusADM	SPEC2006	0.98	3.05
	perlbench	SPEC2006	0.60	0.60
Low	freqmine	PARSEC	0.34	0.34
	gobmk	SPEC2006	0.14	0.20
	fluidanimate	PARSEC	0.14	0.36

5.4 Evaluation Results and Analysis

In this section, I evaluate the performance and energy efficiency for the proposed profiling scheme, and also quantitatively show the effectiveness of two optimization techniques in reducing the profiling overhead.

5.4.1 Memory Access Latency

Figure 16 compares the average memory write latency across different schemes, with the results normalized to `BL`. On average, by applying the proposed techniques step by step, I observed the significant write latency reductions by 19.8%, 37.2% and 63% for `LRS`, `CMP` and `PROF`, respectively. Compared to `RA`, the proposed scheme `PROF` shows 53.5% more reduction. In summary, it is effective to reduce RESET latency by exploiting the number of LRS cells along bitlines.

Since the selective profiling does not change the RESET latency, `SEL_PROF` has the same write latency as that in `PROF`. Since the fine-grained profiling technique may over-estimate the RESET latency, `FINE_PROF` and `SEL_FINE_PROF` exhibit 7.1% write latency degradation over `PROF`. They still achieve 60.3% write latency reduction over `BL`.

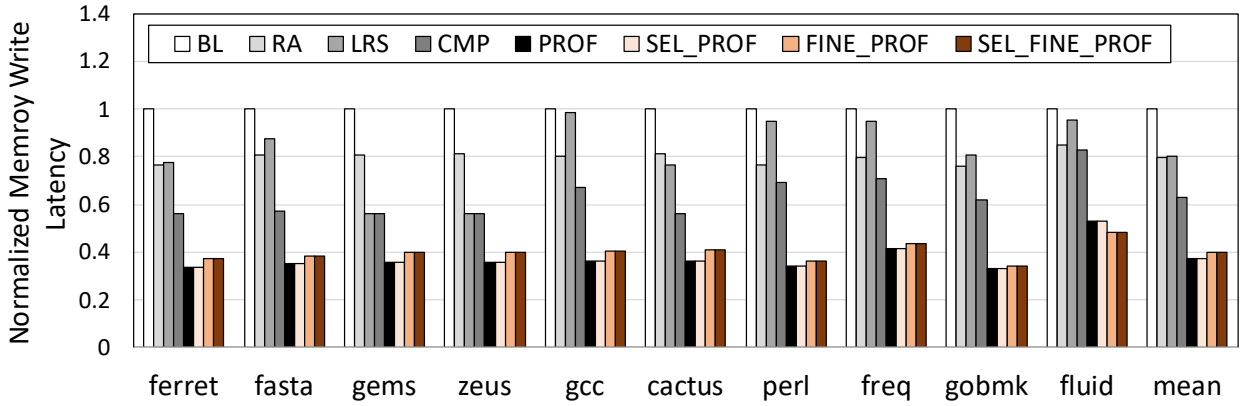


Figure 16: The comparison of memory write latency.

The reduction of RESET latency leads to the reduction of memory read latency. Figure 17 summarizes the memory read latencies in different schemes. The results are normalized

to BL. Similar to the write latency, the memory read latency is reduced by 6.7%, 19.6% and 38.2% for LRS, CMP and PROF respectively. The proposed PROF scheme shows a 27.6% more reduction over RA.

When there are fewer mats profiled with selective profiling, the average profiling latency is shortened and hence the memory access latency on critical path is also reduced. The write latency of SEL_PROF is reduced by up to 39.2% from the baseline. With the fine-grained profiling techniques, FINE_PROF and SEL_FINE_PROF perform slightly worse than PROF. They achieve 36.1% and 37.4% read latency reduction, respectively, over BL.

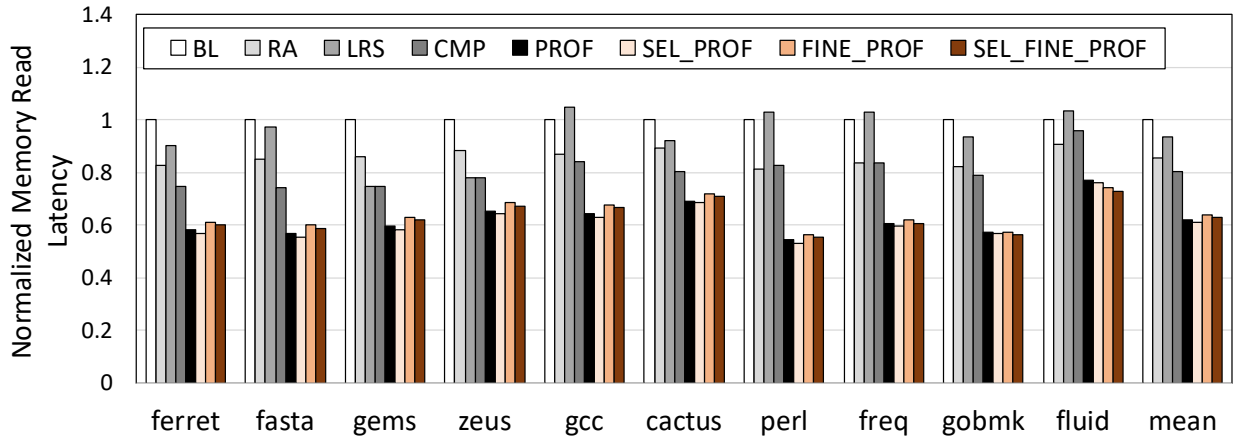


Figure 17: The comparison of memory read latency.

5.4.2 System Performance

I compared the performance when adopting different schemes and summarized the CPI (cycles-per-instruction) results in Figure 18. The results are normalized to BL. From the figure, the proposed profiling schemes achieve larger performance improvements on write intensive benchmarks, e.g., ferret and fasta_dna. On average, PROF outperforms BL by 32.4%, 16.5% and 5.2% on high, medium and low memory intensity benchmarks, respectively. This is because the proposed technique focuses on improving write performance, which is sensitive to the intensity of write requests. On average, PROF achieves 20.5% and 14.2% performance improvements over BL and RA, respectively. Due to shortened profiling operation

latency, SEL_PROF improves the overall performance by 1% over PROF, 21.2% performance improvement over BL. FINE_PROF and SEL_FINE_PROF improve CPI by 18.8% and 19.5%, respectively, over BL.

To illustrate the effectiveness and performance overhead of the profiling techniques, I also compared the proposed designs with IDEAL_PROF, the scheme adopting ideal profiling, i.e., the profiling operation is assumed to have zero latency and not incur any performance overhead. The experimental results showed that, on average, IDEAL_PROF achieves 2% better performance than PROF, and 1.1% better than SEL_PROF. For the group of high memory intensive benchmarks, the average improvement is 3.3% over PROF. From the results, the profiling introduces small performance overhead. Further optimizations, e.g., hiding the profiling latency by issuing profiling commands only during memory bank idle time, are applicable but tend to achieve limited performance improvement with increased hardware cost.

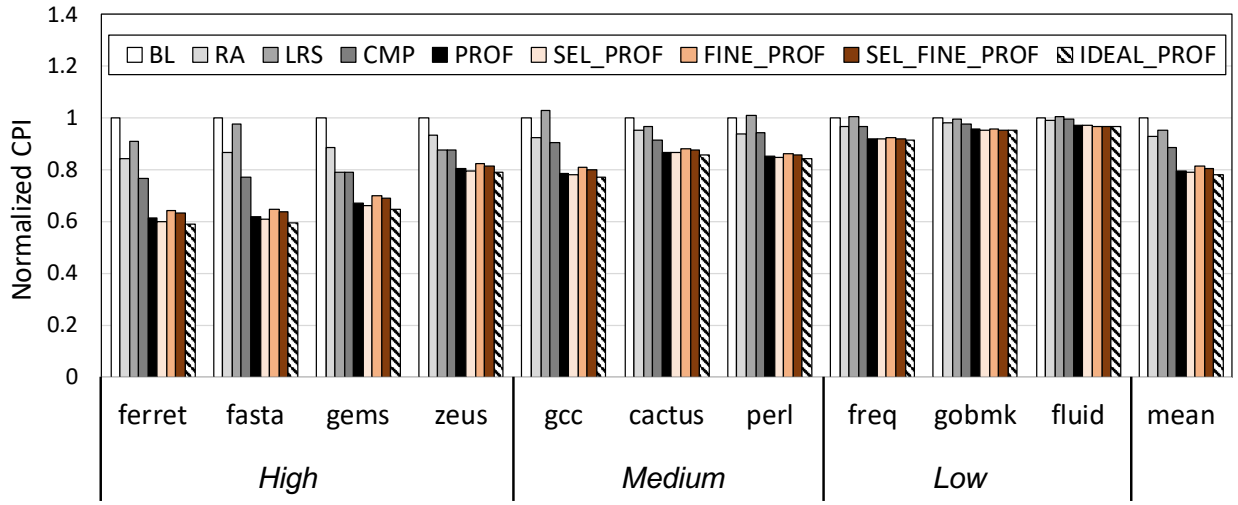


Figure 18: The performance comparison. The benchmarks are categorized into High, Medium and Low memory intensity types based on RPKI and WPKI.

5.4.3 Effectiveness of Profiling Optimization

I next conducted experiments to study the effectiveness of the proposed profiling optimization techniques. The normalized number of profiling operations is reported in Figure 19 and the normalized profiling energy consumption is presented in Figure 20.

Figure 19 compares the number of profiling operations under different optimizations. The results are normalized to `PROF`. On average, `SEL_PROF` i.e., the one adopting selective profiling, reduces 40.6% of profiling operations, while `SEL_FINE_PROF`, i.e., the one adopting both optimizations, reduces the number of profiling operations by 46.3%.

Figure 20 compares the profiling energy with different optimizations. The experimental results show that both optimizations are effective in reducing dynamic energy caused by profiling. By adopting the selective profiling technique, `SEL_PROF` mitigates the energy consumption by reducing the number of profiling operations, while `FINE_PROF` reduces the profiling energy from reading fewer wordlines. From the figure, `SEL_PROF` saves the profiling energy by 40.6% while `FINE_PROF` consumes 93.4% of the profiling energy in `PROF`. The scheme `SEL_FINE_PROF` combines two optimizations and saves 49.9% of the profiling energy in `PROF`.

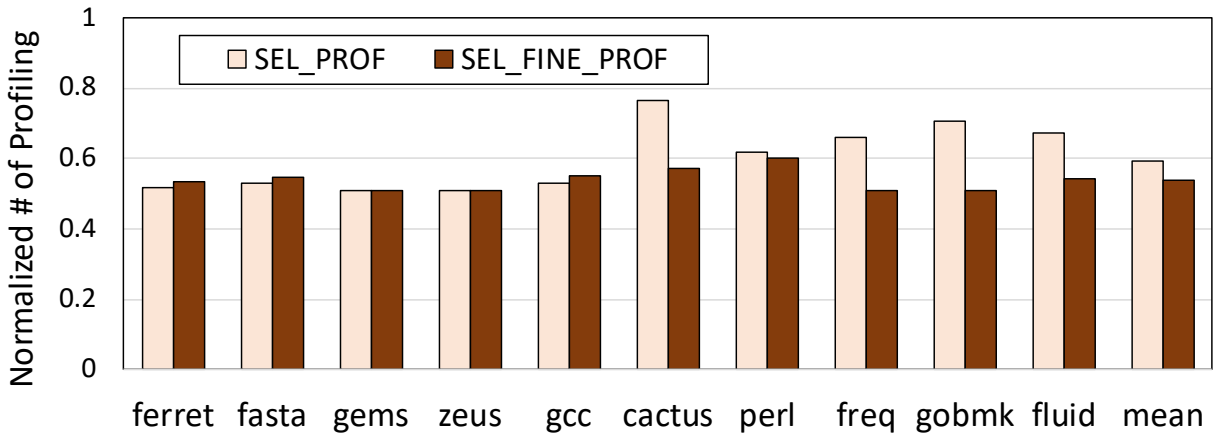


Figure 19: The number of profiling operation performed with optimized techniques on mats (Normalized to `PROF`).

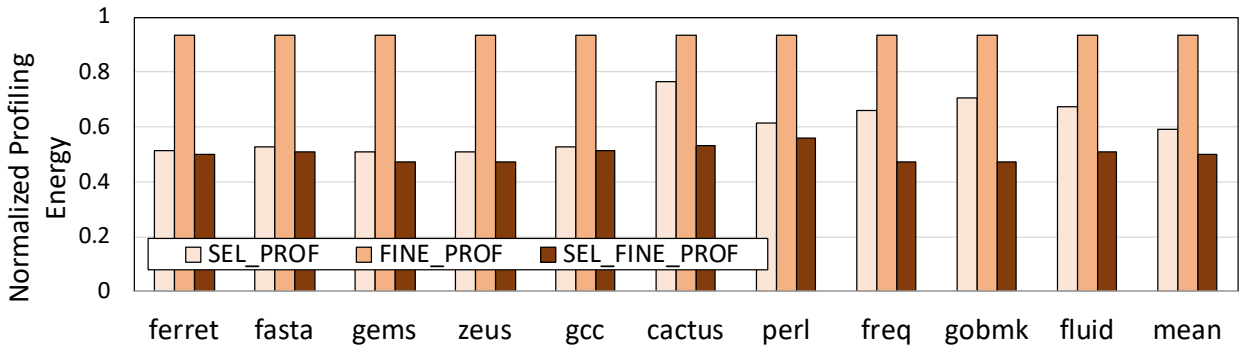


Figure 20: The profiling energy with optimized techniques (Normalized to PROF).

5.4.4 Memory Energy Efficiency

I next compared the dynamic memory energy consumption and energy-delay product (EDP) for all schemes. The results are normalized to BL and summarized in Figure 21. The dynamic energy consumption has three major sources: read, write (including SET and RESET) energy and profiling overheads from my proposed schemes. While the PROF greatly improves RESET performance, it has no impact on read and SET operations. In addition, my proposed schemes introduce profiling overheads. For example, LRS consumes 3.9% more dynamic energy due to the profiling overhead. However, SEL_PROF, FINE_PROF and SEL_FINE_PROF with proposed optimization techniques can reduce the profiling energy effectively as aforementioned.

In summary, PROF achieves 15.7% and 7.6% dynamic energy reduction over BL and RA, respectively, while SEL_PROF, FINE_PROF and SEL_FINE_PROF with optimization techniques further reduce the profiling overhead and achieve 20.2%, 15.4% and 20.3% dynamic energy reduction over BL, though the fine-grained profiling marginally increases write energy. SEL_PROF, FINE_PROF and SEL_FINE_PROF also reduce more dynamic energy than RA by 12.5%, 7.2% and 12.6%, respectively. The EDP results show that the proposed designs can effectively improve the energy efficiency — PROF achieves 31.9% and 19.5% EDP improvements over BL and RA, respectively, while SEL_PROF, FINE_PROF and SEL_FINE_PROF respectively achieve 35.9%, 30.5% and 35.0% EDP improvements over BL. In addition, the schemes SEL_PROF,

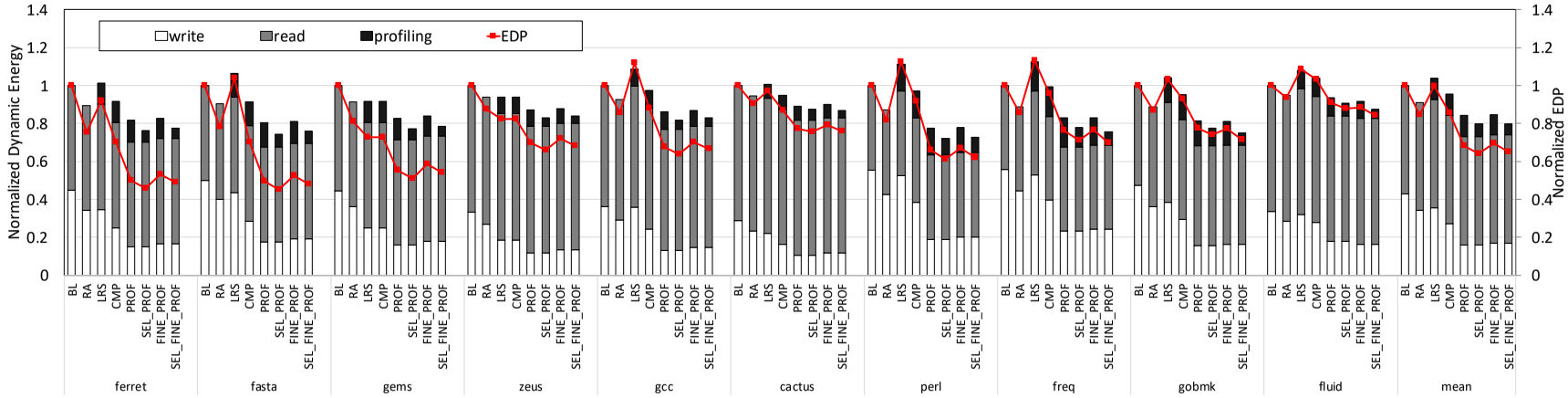


Figure 21: The comparison of dynamic energy and Energy-Delay Product (EDP).

FINE_PROF and SEL_FINE_PROF also outperform RA in EDP improvements by 24.2%, 17.8% and 23.2%, respectively. It is worth noting that though SEL_PROF presents slightly better EDP than SEL_FINE_PROF due to the RESET latency overestimation by adopting fine-grained profiling, the SEL_FINE_PROF can save bandwidth on memory bus by reducing the number of profiling commands and flag bits that represent data patterns, which will eventually reduce bus congestion and save energy consumption on the memory bus.

5.4.5 Sensitivity Study

In this section, the performance and energy efficiency results are finally compared for all proposed schemes with different number of ADC units used in each bank as well as varied ReRAM mat sizes, which are summarized in Figure 22.

5.4.5.1 Sensitivity to Number of ADC units. For the given 512×512 ReRAM crossbar, increasing the number of ADC units can help reducing the profiling overhead. When doubling the number of ADC units from 8 to 16, I summarized the performance improvement and energy reduction results for scheme PROF in Figure 22a. From the figure, while the profiling area and power consumption overhead are doubled, the performance improvements are trivial — only 1.1% improvement was observed. Similarly, SEL_PROF, FINE_PROF and SEL_FINE_PROF cannot significantly benefit from more ADC units.

5.4.5.2 Sensitivity to Mat Sizes. Figure 22b reveals the sensitivity study results when different ReRAM crossbar mat sizes are used — I compare 256×256 and 512×512.

For 256×256 ReRAM mat, the proposed scheme PROF achieves smaller improvements due to smaller IR drop in the array — it has 14.9% performance improvement and 4.6% memory dynamic energy reduction over BL. For the default 512×512 ReRAM mat, the improvements are much larger. In the figure, the proposed scheme PROF is slightly worse (only 1.6%) than RA for 256×256 mat size. This is because the profiling latency and power consumption are independent of mat size, which has a larger impact on smaller mats. The schemes SEL_PROF, FINE_PROF and SEL_FINE_PROF with profiling optimization techniques for 256×256 mat size

reduce dynamic energy roughly to the same extent that they do for 512×512 ReRAM mat. In summary, I expect my proposed designs can achieve larger improvements in future ReRAM arrays that have increasing mat size due to fast technology scaling.

5.5 Conclusion

In this chapter, based on the observation that the RESET latency strongly correlates to the number of cells in low resistant states (LRS) along bit lines, I propose a novel profiling-based ReRAM design, which can exploit the discrepancy of RESET latency. The in-memory processing capability of ReRAM is leveraged to implement a low overhead runtime profiler. By dynamically detecting the number of LRS cells, RESET timing is dynamically adjusted, and significant performance and energy consumption improvements are also achieved. In addition, in order to mitigate the profiling overhead, two optimization techniques — selective profiling and fine-grained profiling, are presented. They both effectively achieve significant profiling energy reduction by reducing the number of profiling operations and halving the number of being read wordlines during a profiling operation respectively. The experimental results show that, on average, my designs improve system performance by 20.5% and 14.2%, and reduce memory dynamic energy by 15.7% and 7.6%, compared to the baseline and the state-of-the-art crossbar design. With all proposed optimization techniques, my design can further reduce dynamic energy by up to 20.3% and 12.6% compared to the baseline crossbar design and state-of-the-art ReRAM crossbar design, respectively.

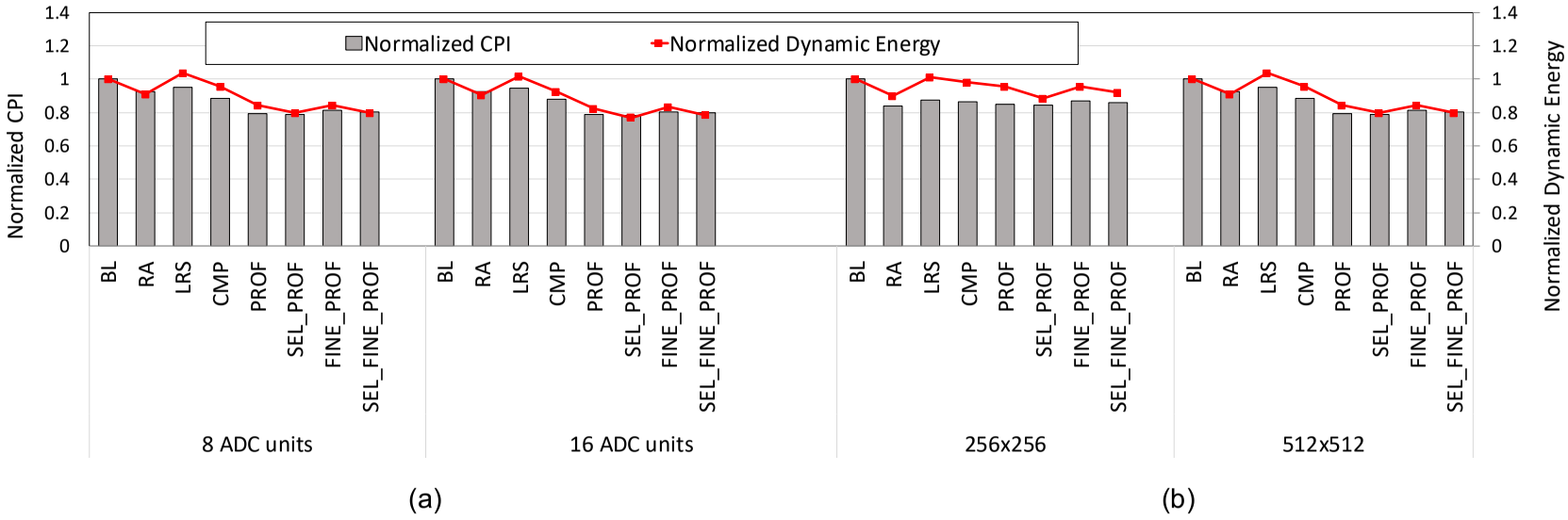


Figure 22: The sensitivity of performance and memory dynamic energy consumption when using (a) different numbers of ADC units; and (b) different ReRAM mat sizes.

6.0 Improving Write Endurance

6.1 XWL: Wear Leveling for Crossbar ReRAM Memory

In this section, I first present an overview of XWL, a table-based wear leveling scheme for ReRAM, and then discuss its design details.

6.1.1 An Overview

The workflow of XWL follows typical table-based wear leveling schemes, which consists of three stages: prediction, address remapping & data swapping and running, as shown in Figure 23. These three stages repeat in every interval, i.e., a number of writes.

XWL splits the whole ReRAM space into chunks and tracks writes to each chunk. In this work, one chunk is a page. Two addresses are differentiated in the following discussion. **Physical address (PA)** refers to the address after OS page table mapping. **Raw address (RA)** refers to the device address where the data are actually saved. As shown in Figure 23, XWL attaches one *interval* entry to each PA chunk and one *lifetime* entry to each RA chunk.

In prediction stage, XWL tracks the number of writes to each PA chunk in the corresponding *interval* entry and the number of lifetime effective writes to each RA chunk in its *lifetime* entry. The major difference between XWL and conventional wear leveling is, instead of tracking raw write accesses for both tables, XWL records effective writes to update the $lifetime$ table and *raw writes* to update interval write table.

In address remapping & data swapping stage, XWL chooses one RA chunk and one PA chunk that are not mapped to each other. The choice involves two pairs, their PA to RA mapping are changed accordingly. For example, in Figure 23, if PA-chunk-2 and RA-chunk-1 are chosen, since PA-chunk-1 maps to RA-chunk-1, and PA-chunk-2 maps to RA-chunk-2, the swap results in PA-chunk-1 maps to RA-chunk-2 and PA-chunk-2 maps to RA-chunk-1, as shown in the figure. The candidate selection policy determines what pages are chosen to

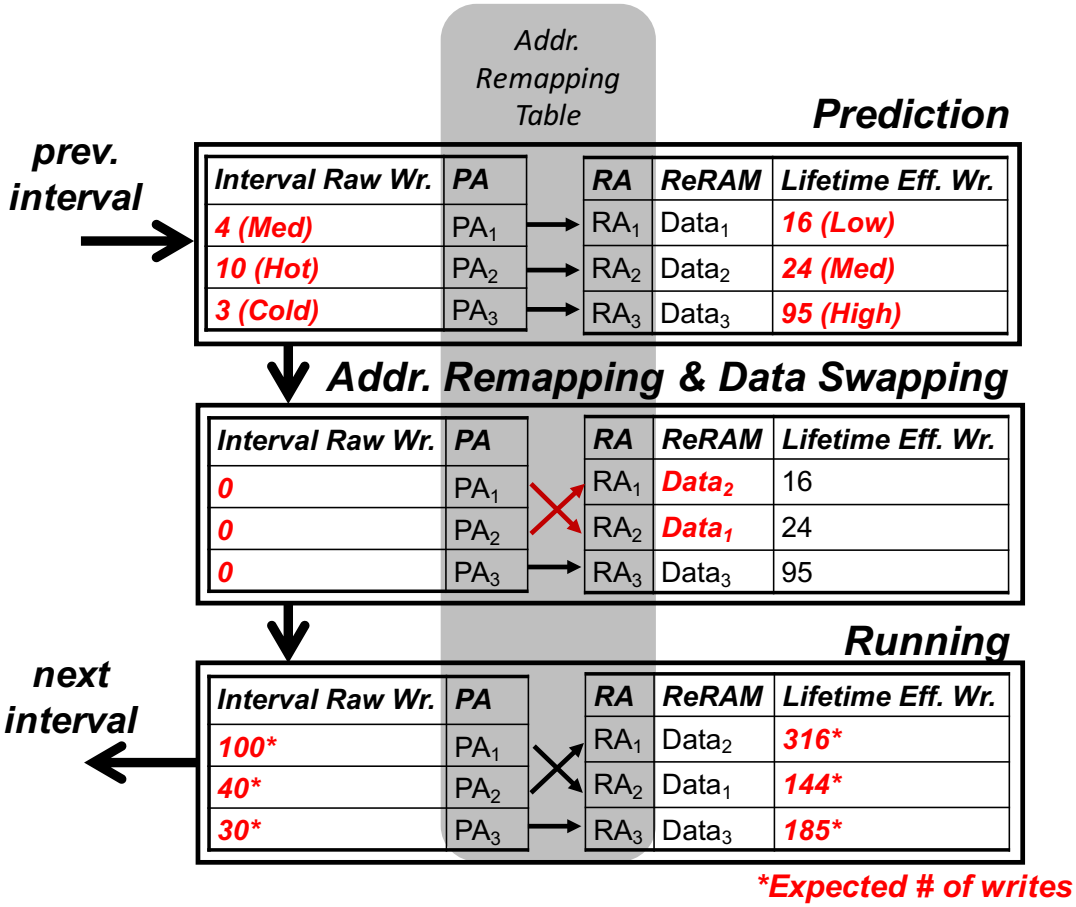


Figure 23: The basic workflow of XWL.

get remapped. I will present different algorithms in the next subsection. The design is to map hot physical pages to the ReRAM pages with the least degree of wearing out, similar to those previously design table-driven wear leveling algorithms [117]. Remapping involving reading two blocks and write two blocks. Clearly, the bigger the chunk is, the larger overhead the swap is. XWL cleared the interval entries after the swap.

In the running stage, each ReRAM page tracks incoming writes with predicted distribution, matching hot pages to low wearing out domains and cold pages to high wearing out domains, which achieves the aim of enhancing lifetime for overall crossbar ReRAM memory. During the running phase, both of two write tables keep updating with new write operations.

6.1.2 Design Details

6.1.2.1 Effective and Raw Write In the proposed XWL scheme, both of the number of effective writes and raw writes are tracked at runtime. The effective write total of each chunk indicates how much lifetime the corresponding chunk has experienced while the raw write reflects the intrinsic access patterns of applications. The raw count would not change if having PA chunk remapped to a different RA location. However, their effective write counts depend on mapping. The number of raw writes in each interval is used to indicate how many incoming writes will reach to each ReRAM page. In contrast, to determine the degree of wearing out of each page, the proposed effective write needs to be adopted for lifetime write table since it measures how many more writes each page can undertake before failures.

6.1.2.2 Updating Write Tables While it is straightforward to update the interval raw write table, i.e., increment after each read or write, to update the lifetime table, Equation 4.3 is adopted and the effective write is computed based on the write pulse width. Figure 24 illustrates the profiling scheme which is used for dynamic RESET latency as well as updating effective write table.

As the bitline data pattern profiling and dynamic RESET latency presented in Chapter 5 are adopted, the RESET latency is determined by row addresses and runtime bitline data patterns. In order to ensure the correctness of write timing, I conservatively assume each

write after profiling always introduce one more LRS cell on the worst-case bitline, which prolongs the RESET latency. Similarly, I also have conservative assumption of updating effective writes. However, in contrast to dynamic RESET latency, I assume writes will bring more HRS cells instead, since more HRS cells lead to larger voltage drop on selected cells. Therefore, the worst-case LRS cell ratio has to be tracked to look up dynamic RESET timing as well as worst-case HRS cell ratio needs to be tracked to update effective write table. In the case shown in Figure 24, in one row address group n of a simplified ReRAM crossbar, the worst-case LRS cell number is 5, and the worst-case HRS cell number is 3, both of which are incremented by 1 for each write request after profiling.

The dynamic RESET timing is simply determined by the table shown in Figure 7b, which maps LRS cell ratio in a particular row address group to a conservative RESET timing. For discussion purpose, I assume the RESET latency is t_R in this case. As it is desired to RESET multiple cells, e.g. at most 8 bits in the design, within one ReRAM crossbar, the t_R is most conservative RESET timing to ensure write success, but it is too aggressive to use this latency to estimate effective writes with Equation 4.3. This is since the t_R may be too long for other bits that have larger voltage drop on selected cells owing to more HRS cells on their bitlines. When all bits are RESET with same t_R, those victim cells that take much longer RESET time than ideal one may be over-RESET, which leads to a endurance degradation. Therefore, the most conservative effective writes needs to be calculated by using following formula:

$$EW_{addr} = EW_0 \cdot a \cdot e^{b \cdot (V - V_0)} \tag{6.1}$$

where EW_{addr} is the most conservative effective writes at address $addr$, EW_0 is $\left\lceil (\frac{t_R}{t})^2 \right\rceil$ with RESET timing t_R, V_0 is the voltage drop at the worst-case LRS cell ratio, and V is the one at worst-case HRS cell ratio, and a, b are fitting constants. Equation 6.1 is derived from experimental data of the different over-RESET voltages with same RESET pulse width on endurance degradation in [14].

6.1.2.3 Address-Remapping Algorithm As write tables are updated for each interval in memory controller, the physical addresses from CPU need to remap to ReRAM page real addresses while migrating data accordingly. As evaluated in experiment section of this

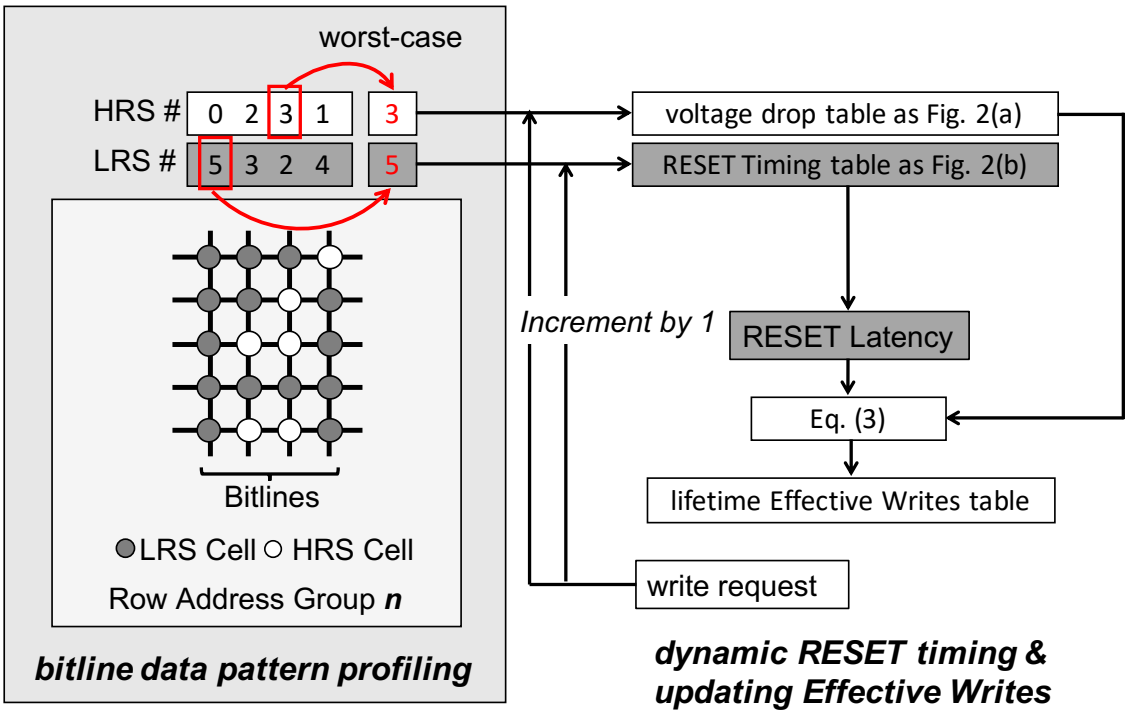


Figure 24: Profiling bitline data pattern for (1) optimized RESET latency and (2) estimating effective writes.

chapter, the naïve wear leveling technique, which simply remaps PA with largest raw writes to RA with smallest number of effective writes, helps to improve lifetime of ReRAM crossbars to certain extent. However, obviously this scheme ignores the fact that all pages are not worn out equally, and they actually depend on dynamic bitline data patterns and physical locations. Therefore, with only taking write access patterns of applications into consideration, it may be not able to effectively leverage incoming writes after address remapping.

In addition to raw write access patterns, I also want to exploit the impact of ReRAM crossbar features on endurance for address remapping. The *weight* is introduced to indicate the tendency of remapping a PA to a physical ReRAM crossbar page. Figure 25 illustrates my address remapping scheme. In this example, I partition ReRAM crossbar into 5 address groups. According to preceding discussion, the closer the group is from the write drivers, the more stress its cells accumulate from each write. Therefore, each group is assigned a different weight as follows.

$$weight_{addr} = \frac{\sum_{r=0}^{n-1} EW_{addr}^{r}}{n} \tag{6.2}$$

where $weight_{addr}$ is the weight at page address $addr$ and EW_{addr}^{r} is the effective writes at page address $addr$ with LRS cell ratio of r. It is worth noting that effective writes are averaged at same address with n different LRS cell ratios. This is since the data pattern can significantly change after prediction with much longer interval (10^4 writes) than profiling (64 writes), and it is no longer feasible to exploit bitline data pattern to estimate actual wearing out for future writes.

Moreover, I adopt the *Predict Write*, which estimates upper limit of effective writes if all writes reach to a particular page. It can be calculated by following equation:

$$PredictWr_{addr} = EW_{addr} + weight_{addr} \times interval \tag{6.3}$$

where $PredictWr_{addr}$ is the *Predict Writes* at page address $addr$ and $interval$ is a parameter of how many writes between an address remapping.

Finally, as Figure 25 shown, the PA with the largest number of raw writes remaps to RA with smallest *predict writes* instead of effective writes, and vice versa.

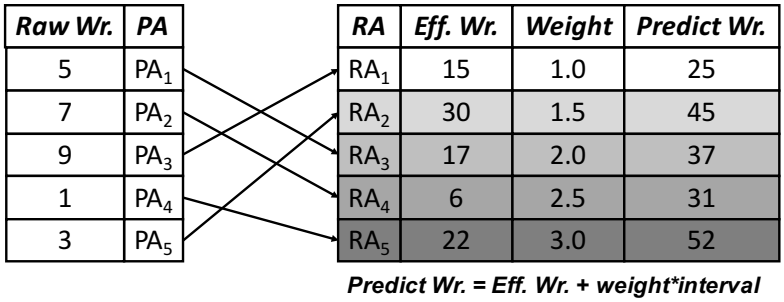


Figure 25: An example of PA to RA address remapping.

6.1.2.4 Design Overhead XWL adds two tables with two entries per 4KB data chunk — 20 bits and 14 bits are used for the *effective writes* and *interval raw writes* counter, respectively. One 16-bit remapping entry is added for each chunk. The total space overhead is approximately $50bits/4KB = 1.56 \times 10^{-3}$. I assume the optimized write scheme exploits the LRS cell ratio information. If not, adding online profiling introduces negligible overhead, as shown in Chapter 5. I use CACTI [66] to model the two tables as direct mapped cache, the area and energy overheads are also negligible.

6.1.3 Process Variation Issue

Process variation (PV) is not considered in this work. When taking PV into consideration, some of cells/rows would be more vulnerable to write operations than others. Several PV aware wear leveling techniques [19, 123, 117] have been recently proposed to mitigate this issue. XWL is a table based wear leveling scheme, which has the ability to address PV more flexibly. These designs are orthogonal to XWL in the work.

6.2 Experimental Setup

In Chapter 4, I model and simulate a 512×512 ReRAM crossbar to investigate the correlation between RESET latency and *effective writes*. In addition, I used an in-house

architectural Chip Multiprocessor simulator to evaluate the proposed XWL scheme and compare it with baseline as well as naïve design. The system configuration is presented in Table 7. Pintool [61] is used to collect memory access traces from PARSEC [5], BioBench [3] and SPEC2006 [31] benchmark suites. All benchmarks are executed with or without wear leveling until first ReRAM page is worn out. *Flip-n-write* [17] is also used to reduce the number of written bits. With a representative ReRAM device, I assume the ReRAM cell endurance is 1.6×10^6. For the proposed XWL, the default interval is 10^4 while different intervals are also evaluated in experiments. The benchmarks are characterized in Table 8 with write bandwidth to ReRAM memory. I adopt the profiling approach and dynamic RESET latency from Chapter 5.

In the work, I compared the following wear leveling schemes:

- `NoWL`: baseline scheme, which adopts dynamic RESET latency and data pattern profiling, does not use any wear leveling techniques.
- `Naïve`: the wear leveling scheme, which follows the workflow introduced in Section 6.1.2, does not use proposed address remapping algorithm.
- `XWL`: the proposed wear leveling design.

Table 7: System configuration

Processor	4 cores@1.8Ghz; single issue in-order CMP
L1 I/D-cache	Private; 16KB/core; 4-way; 2 cycles
L2 cache	Private; 1MB/core; 8-way; 64B; 10 cycles
Main memory	2Gb ReRAM; 4KB page; 64B per line; 1 rank; 8 chips/rank;8 banks/chip; 128 mats/bank;
ReRAM Timing	Read Latency 18ns@1.5V; SET latency 10ns@3V; RESET latency based on profiling@-3V

Table 8: Benchmark summary

Name	Benchmark Suite	Write Bandwidth to ReRAM (MBps)
ferret	PARSEC	139.0
fasta_dna	BioBench	129.4
GemsFDTD	SPEC2006	123.2
bzip2	SPEC2006	61.3
zeusmp	SPEC2006	60.8
gcc	SPEC2006	56.6

6.3 Evaluation Results

6.3.1 Endurance Improvement

Figure 26 presents the endurance improvements (normalized to `NoWL`). On average, by applying the proposed wear leveling techniques, I observed the significant endurance improvements by 285% and 324% for `Naïve` and `XWL`, respectively. Moreover, the proposed wear leveling `XWL` shows 14% more lifetime enhancement. In conclusion, by using proposed concept of *effective write* as well as the address remapping algorithm, the lifetime of crossbar ReRAM memory is effectively improved.

To evaluate the impact of interval length, Figure 27 compares the normalized endurance improvements with different intervals, i.e., 10^4, 5×10^4 and 10^5. From the figure, the effectiveness of endurance improvement diminishes as interval gets longer for most benchmarks. On average, the normalized endurance improvements by using `XWL` with intervals of 10^4, 5×10^4 and 10^5 are 324%, 216% and 166%, respectively. This indicates that the proposed `XWL` can still significantly improve the endurance of crossbar ReRAM memory even with longer address remapping intervals.

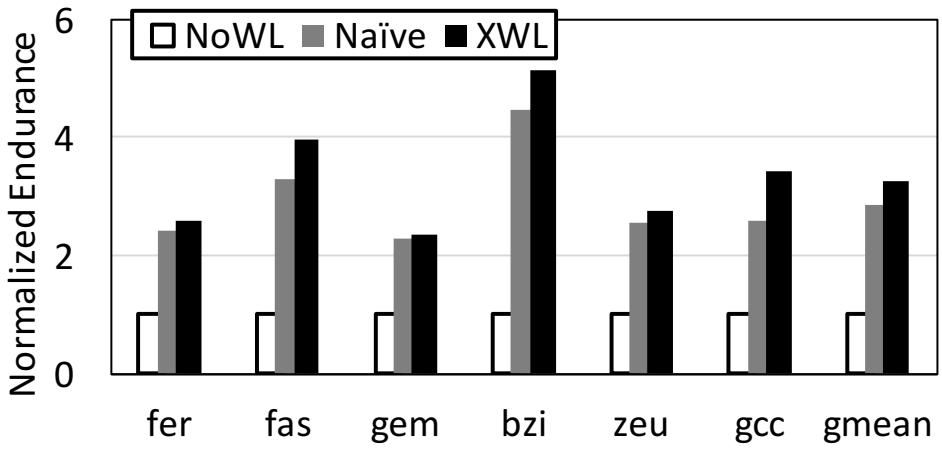


Figure 26: Comparison of normalized endurance.

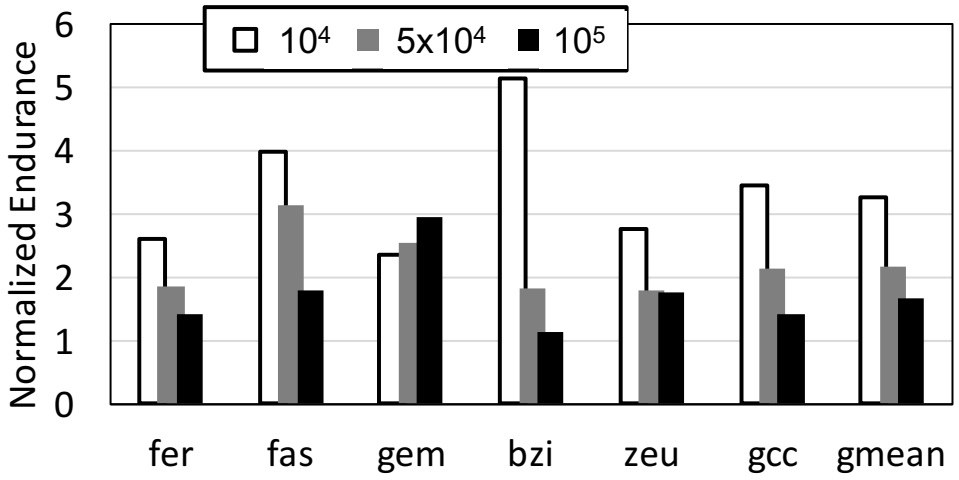


Figure 27: Comparison of normalized endurance with different remapping intervals.

6.3.2 Performance Overhead

The data swapping after address remapping is inevitable for wear leveling, while it also contributes major performance overhead [19, 117]. I also evaluate the performance overhead of introducing the proposed wear leveling techniques. Figure 28 shows the swapping overhead in performance by using `Naïve` and `XWL` designs. The swapping overhead is defined as follows:

$$Swapping \;\; Overhead = \frac{t_{data\_swapping}}{t_{execution}} \tag{6.4}$$

where $t_{data\_swapping}$ and $t_{execution}$ represent total data swapping time and execution time in cycles through whole memory system lifetime, which indicates the overall percentage of ReRAM crossbar lifetime are used for data migration. Overall, `Naïve` and `XWL` incur 6.5% and 6.1% performance overheads respectively. Though the `XWL` may potentially result in less hot ReRAM pages write to the rows with smaller RESET latency as well as a larger number of data swapping through the whole system lifetime, its performance loss is slightly better than `Naïve` since the `XWL` can much better improve the endurance cycles than `Naïve`.

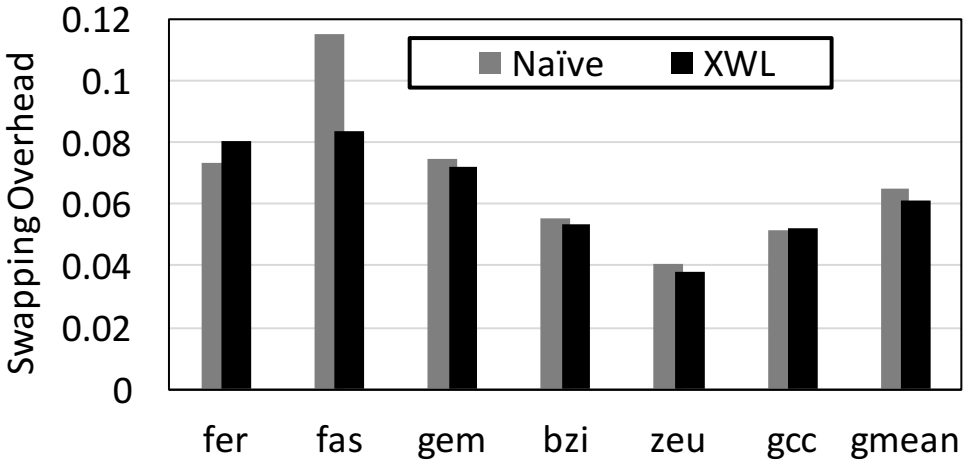


Figure 28: Comparison of data swapping overhead.

6.4 Conclusion

In this chapter, I focus on mitigating the write endurance degradation from IR drop by proposing a novel wear leveling scheme for crossbar ReRAM memory. Specifically, based on the study the write endurance variation issue in crossbar ReRAM memory in Chapter 4, in which I observe that the effective write, which indicates actual the degree of ReRAM wearing out, dynamically changes in runtime with different data patterns and row addresses, I propose a novel wear leveling scheme based on effective write to enhance lifetime of crossbar ReRAM memory. To the best of my knowledge, this work is the first study specifically on addressing the write endurance issue for crossbar ReRAM memory. The final evaluation results reveal that, my design improves write endurance by 324%, compared to the baseline design.

7.0 Enhancing Lifetime for ReRAM Crossbar Based Neural Network Accelerators

7.1 Background

In this section, I discuss about the ReRAM crossbar array and its applications as neural network accelerators, and also briefly introduce the neural network training.

7.1.1 ReRAM Crossbar and Its Application for Neural Network Computing

Figure 29 illustrates an ReRAM crossbar architecture, in which each ReRAM cell is connected to a worldline and bitline at their crosspoint. With a voltage stress, ReRAM cell behaves as resistive devices obeying Ohm's law. Hence, the current flowing through each cell depends on its resistance and voltage stress. With a vector of n input voltages $\mathbf{V} = [V_0, ..., V_{n-3}, V_{n-2}, V_{n-1}]$ from wordlines to one particular column of ReRAM cells, as highlighted in red in Figure 29, aggregated analog current $I = \sum_{n=0}^{n-1} V_i \cdot G_i$ outputs from the bitline, where G_i is the conductance (the reciprocal of resistance, $G_i = 1/R_i$) of the ReRAM cell. If the voltage $\mathbf{V}$ and conductance $\mathbf{G}$ are treated as input vectors, the output $I = \mathbf{V} \cdot \mathbf{G}$ is naturally a result from a mathematical dot-product calculation by $\mathbf{V}$ and $\mathbf{G}$. Since such dot-product operations are predominantly performed in neural network computing, with weight matrices represented by different resistance levels in ReRAM cells, they can be efficiently processed inside ReRAM crossbars.

7.1.2 Neural Network Training

Figure 30 shows an example of neural network training, which is composed of a forward and a backward propagation. In forward propagation, an input vector $[x_0, x_1, ..., x_n]$ is fed into the network while calculating the intermediate neurons with weight matrices $W_1, W_2, ..., W_4$ in each layer. Afterwards, an output vector $[y_0, y_1, ..., y_m]$ is computed and taken by a *loss function* to estimate the difference with labeled data. As soon as the loss is

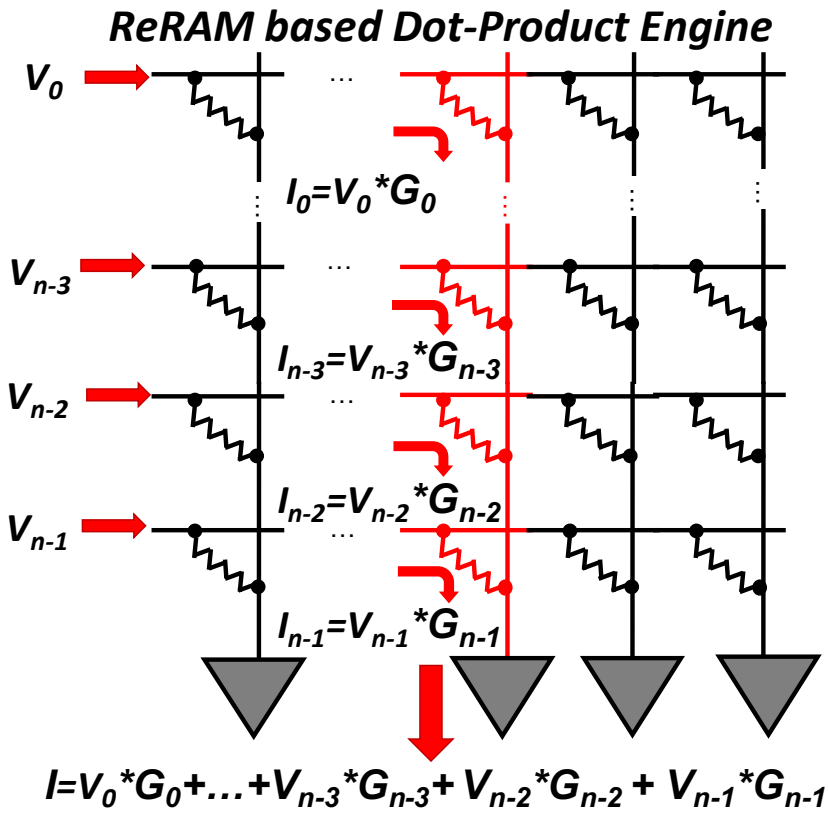


Figure 29: An ReRAM crossbar based dot-product engine.

obtained, a backward propagation starts by sending the loss back to all layers of the neural network. During this stage, weight matrices are frequently updated with the loss by using: $\Delta \mathbf{W_i} = -\mathcal{LR} \cdot \frac{\partial Loss}{\partial \mathbf{W_i}}$, where $\Delta \mathbf{W_i}$ is the update to each of weight matrix, $\mathcal{LR}$ is the learning rate, and $i = 1, 2, ..., 4$ denotes the number of the layer.

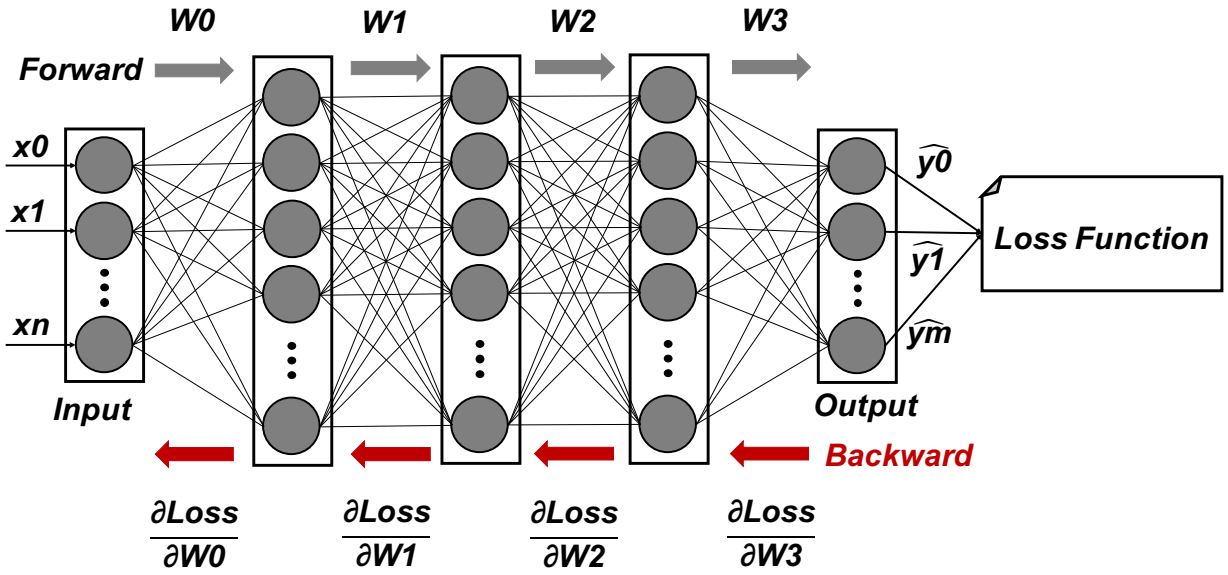


Figure 30: Neural network training with weight updates.

7.2 Motivation

In this section, I analyze the ReRAM wearing out mechanism and stochastic switching behaviors, which lead to proposing innovative solutions for mitigating endurance degradation of ReRAM crossbars during the training.

7.2.1 ReRAM Cell Endurance Model

The wearing out mechanism of ReRAM cell has been long studied [69, 86, 14, 34], which all generally believe excessive programming conditions, such as long programming pulse width and strong pulse amplitude than necessary, i.e., over-SET/-RESET, degrade ReRAM

cell endurance. In order to analytically model ReRAM endurance degradation, it is necessary to identify the key factors that can limit the write cycles of an ReRAM cell.

7.2.1.1 Tunneling Gap Distance and R_{off}/R_{on} Figure 31 shows the how resistance level is determined during ReRAM cell switching. As discussed in Chapter 2, a formation/rupture of CFs in an ReRAM cell happens during SET/RESET processes. For an instance of RESET process shown in Figure 31, with a negative voltage stress on top electrode, the CFs are dissolved. The stronger RESET condition can lead to less amount of residual CFs, and thereby the cell exhibits a larger resistance. Based on the ReRAM cell model presented in previous work [110, 13, 40], the concept of tunneling gap distance g, which denotes an average distance from the top of residual CFs to the top electrode layer, is used to indicate the resistance level of an ReRAM cell during switching. An $I-V$ characteristic equation in an ReRAM cell can be represented as $I = I_0 exp(-g/g_0) sinh(V/V_0)$, where I_0, g_0 and V_0 are fitting constants [13]. In the figure, a tunneling gap $g2$ is larger than $g1$, which implies that a stronger programming condition is needed for switching the cell to $g2$ than $g1$. Consequently, an ReRAM cell with a tunneling gap $g2$ has a larger resistance than the one with $g1$.

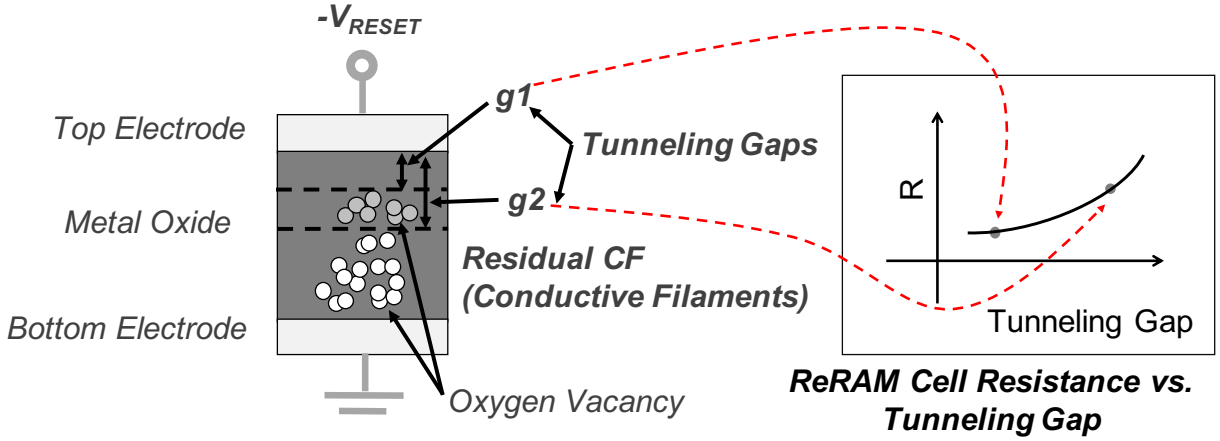


Figure 31: ReRAM cell switching and its resistance.

Recent studies [103, 59] report that the SET process is abrupt and RESET process is more gradual, and prior studies [110, 105, 114, 99] also present that RESET takes much

longer time and consumes much more energy than SET operation. Therefore, in this work, I assume SET operation is fast and accurate without consuming much energy, and the endurance degradation principally comes from RESET operation. I also assume that each SET operation accurately switches the cell to R_{on}, and thus the R_{off}/R_{on} ratio is determined by RESET operation condition. However, it is worth noting that my proposed schemes are also applicable to different ReRAM switching assumptions, such as symmetric SET/RESET operations. With a fixed or variable R_{off}/R_{on} ratio, the relationship between endurance degradation and programming strategies are different, which consequently results in different endurance enhancement solutions.

7.2.1.2 Fixed R_{off}/R_{on} During Programming With a fixed R_{off}/R_{on} ratio, recent studies [86, 115] reveal a tradeoff between endurance and programming latency that a longer programming pulse without over-RESET the cell can prolong the cell endurance. The hypothesis of this argument is to switch an ReRAM cell to a fixed resistance level, that is to say, the g is unchanged under different switches [86]. The tradeoff of endurance and write latency can be approximated as: $Endurance \approx (t_W/t_0)^C$, where t_W is the write latency and t_0 and C are fitting parameters. The same $C = 2$ is used as [115] in this work. Based on this observation, a concept of *effective write* is proposed in Chapter 4 to estimate the endurance degradation in ReRAM crossbars with taking sneak current issue and RESET latency discrepancy into consideration. As reported in Chapter 6, it is necessary to adopt optimal RESET latency at runtime to avoid excessive write strength. In this chapter, the effective write is adopted as the metric to estimate the degrees of wearing out an ReRAM cell and computed by Equation 4.3.

7.2.1.3 Variable R_{off}/R_{on} During Programming In contrast to a fixed R_{off}/R_{on} scenario where prolonging RESET duration to mitigate endurance degradation can be used, with a flexible R_{off}/R_{on}, the endurance is improved in a different approach. A recent research [69] demonstrates that the ReRAM endurance is significantly correlated to R_{off}/R_{on} ratio. The larger R_{off}/R_{on} is, the shorter the lifetime of an ReRAM cell can have. Additionally, the programming pulse width, i.e., RESET latency in this work, is proportional

to R_{off}/R_{on} ratio. Based on above two observations, an analytical model is presented by using data from [69] with $CuTe_x/HfO_2$ material (which has the best endurance and hence is suitable for neural network training) to estimate the lifetime of an ReRAM cell in different RESET latencies:

$$Endurance \approx a \cdot e^{b \cdot WM} \tag{7.1}$$

and

$$WM \approx p_0 \cdot t + p_1 \tag{7.2}$$

where WM denotes the R_{off}/R_{on}, t is RESET latency and a, b, p_0, p_1 are fitting constants. While shortening the RESET latency linearly decreases the R_{off}/R_{on} ratio, the endurance is exponentially improved by a reduced R_{off}/R_{on}. Equation 7.1 and 7.2 together imply that appropriately optimizing RESET latency may help to achieve better endurance.

7.2.2 ReRAM Stochastic Switching

Though with an appropriate RESET condition, the ReRAM cell can be switched to a targeted resistance level by forming a certain tunneling gap g. However, we should also realize that the programming on ReRAM cell is a stochastic switching [55]. Previous studies [55, 25, 84] report that the switching behaviors of an ReRAM cell is stochastic, and its probability is predictable with modeling the correlation between programming conditions and successful switching rate. It is worth noting that, a successful switching rate here is defined as — *how many successful read-out values (SLC reading mode with values '0'/'1') are as expected out of total read attempts under the same read voltage condition* [25], which indicates that a targeted R_{off}/R_{on} should be achieved in order to provide enough read margin. Otherwise, a reduced R_{off}/R_{on} can lead to a uncertain switching.

Two major programming conditions — RESET pulse width (time) and height (amplitude), have significant impact on switching probability. They both in fact affect on R_{off}/R_{on} as discussed before. In this work, following Equation 7.3, as reported in [84] with RESET conditions from Chapter 5, is used to model the correlation between switching probability and RESET time under different pulse heights:

$$P = \frac{1}{2} erfc(-\frac{\ln t_w - \ln \tau}{\sqrt{2\sigma}}) \tag{7.3}$$

where the P is the ReRAM cell switching probability, $erfc(x)$ is a complementary error function, t_w represents RESET pulse width (write latency), τ and σ are fitting parameters.

Figure 32 plots a group of curves with ReRAM cell switching probabilities at different RESET voltage widths and heights. In this work, I assume these optimized RESET latencies from Chapter 5 guarantee a 100% cell switching. When applying a shorter RESET timing than those under the same data pattern, the switching probability is smaller than 100% and can be predictably computed with Equation 7.3.

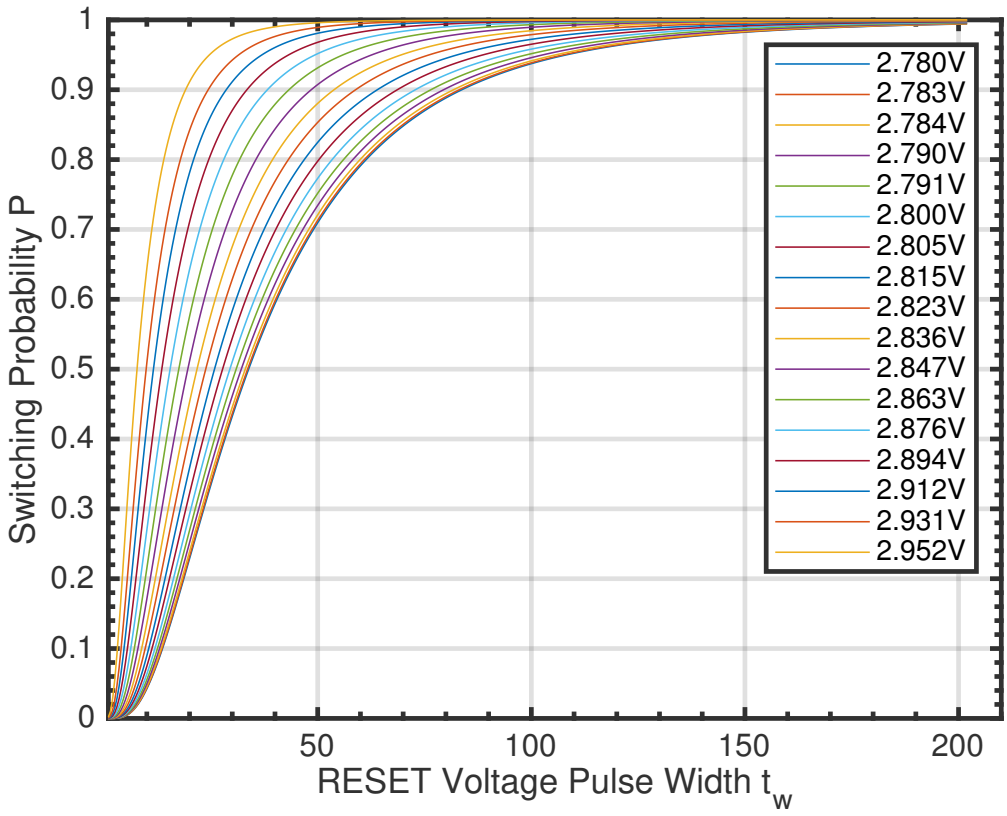


Figure 32: The correlation between switching probability and RESET voltage width with different RESET pulse heights.

7.3 Proposed Designs

In this section, I elaborate the proposed framework, ReNEW, which can effectively improve the endurance of ReRAM crossbar based neural network accelerators.

7.3.1 Training NN with SLC ReRAM

As shown in Figure 33, the ReRAM crossbar based neural network accelerator adopted in this work has a similar architecture to PRIME [16]. This architecture is composed of several banks, each of which further consists of many ReRAM crossbars. The ReRAM crossbars can be partitioned into *memory array* and *compute array* based on their usage. Memory arrays are to store temporary data, while compute arrays primarily perform in-memory dot-product calculations.

Different from most of prior work [9, 73, 85, 116] where Multi-Level Cell (MLC) ReRAM crossbars are adopted for neural network training, I propose to program ReRAM crossbars in Single-Level Cell (SLC) mode for training, but well-trained weight matrices for inference task are still programmed in MLC mode. This is because, compared to MLC ReRAM crossbars, using SLC ReRAM crossbars for neural network training owns following advantages: (1) Since an iterative scheme and a large R_{HRS}/R_{LRS} ratio are usually necessary for MLC ReRAM programming, write endurance of an SLC ReRAM cell can be as much as 4-6 magnitudes better than MLC ReRAM according to prior study [72]. (2) With similar reasons to (1), programming an SLC ReRAM cell also performs better than an MLC ReRAM [106] cell in performance and energy-efficiency. As shown in Figure 33, 8-bit fixed-point numbers are used to represent weight data. It is worth noting that, though I assume bits within the same weight data are stored consecutively in one row, my proposed schemes are also applicable to different weight mapping approaches without significant changes. To store each of this 8-bit weight data, 8 cells in an SLC ReRAM crossbar are required, however, only 2 cells (if 4-bit per cell is assumed) are enough in an MLC crossbar array. Even though MLC ReRAM is several times denser than SLC ReRAM (depends on the resistance levels), the capacity loss is potentially compensated by reloading weights onto SLC ReRAM crossbars for

a few times. A comparison of MLC and SLC ReRAM crossbars for neural network training is presented in the experiment section of this chapter. To enable using multiple cells for representing one weight data, a bit slicing technique [22] is adopted to support dot-product calculations on SLC ReRAM crossbars. In addition, necessary compute units for shift-add operations are also needed to generate final results.

After the completion of neural network training, the trained weights shall be re-programmed into ReRAM crossbar arrays in MLC mode. This is because, during the training, partitioning and reloading weights onto crossbars are possibly needed due to limited capacity of SLC ReRAM crossbars. However, an inference task usually demands near real-time response. Hence, reloading weights with using crossbars in SLC mode may be not suitable for the inference. Besides, writing well-trained weights into crossbars is not a frequent operation during the inference, therefore, the energy and performance cost are more acceptable than during the training.

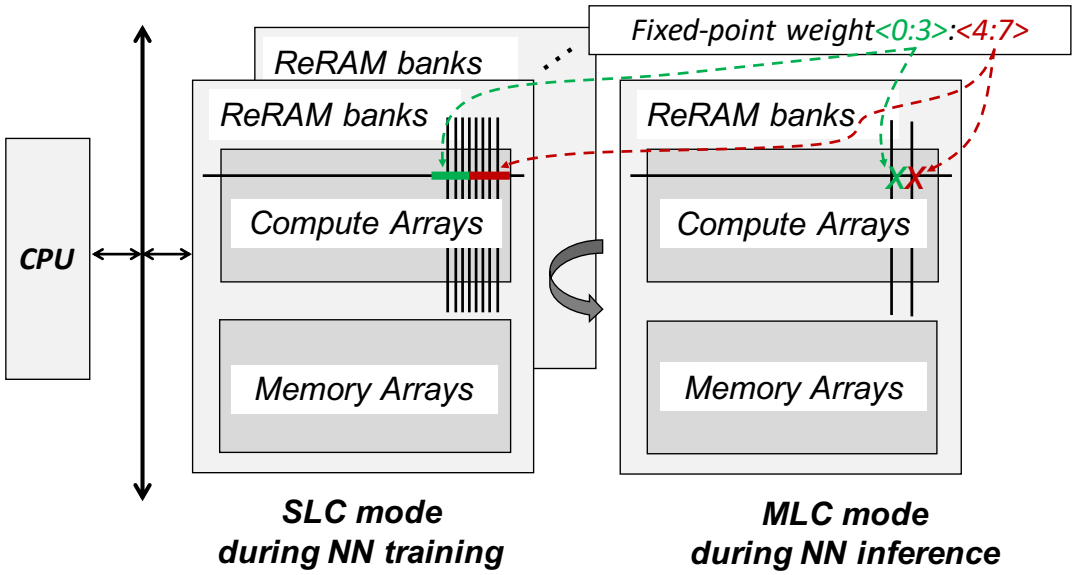


Figure 33: An overview of ReRAM crossbar based accelerator for neural network computing.

7.3.2 Optimized Programming Order

The study in Chapter 6 on write endurance variation shows that having more LRS cells along bitlines benefits the lifetime of ReRAM cells under the premise of no over-SET/-RESET cells. This observation leads to optimizing the weight updates order by programming cells from '0' to '1' (SET operations) before performing RESET operations. Figure 34 illustrates a comparison of a sequential weight update, which is essentially a row-major order, and the proposed optimized programming scheme that first performs SET operations to increase the number LRS cells in ReRAM crossbars, and then RESET the rest of cells.

By employing the optimized programming order, the effective writes brought by each RESET are expected to decrease, since more LRS cells have been generated beforehand. As a side note, this optimization guarantees successful switching since the targeted R_{off}/R_{on} shall be achieved, and it only prolongs endurance as Equation 4.3. Therefore, it can be safely applied to either LSB (least-significant bits) columns or MSB (most-significant bits) [1] columns as it basically does not introduce switching errors. Moreover, this optimized programming order does not introduce latency penalty since separate SET and RESET programming phases are also needed in the baseline row-major order [105].

7.3.3 Shortened RESET operation

As discussed in Section 7.2, by shrinking R_{off}/R_{on}, the lifetime of ReRAM cells can be exponentially improved. Equation 7.2 proves that a shortened RESET latency linearly reduces R_{off}/R_{on}. With observations above, I propose to shorten RESET time on selected columns in ReRAM crossbars, such as those contain LSB, to extend their lifetime.

Though a shortened RESET duration significantly mitigates endurance degradation, in the meantime it potentially brings switching errors, which possibly further results in an accuracy loss. To address this, I propose to exploit the intrinsic error-tolerance characteristic of neural network training, which has been widely reported in prior studies [122, 29], to mitigate the accuracy loss. Intuitively, it is possible to achieve a sweet spot in the tradeoff between

[1] In this work, I denote MSB/LSB as the most/least significant half in each weight data, e.g., 4 most-significant bits and 4 least-significant bits for an 8-bit weight number.

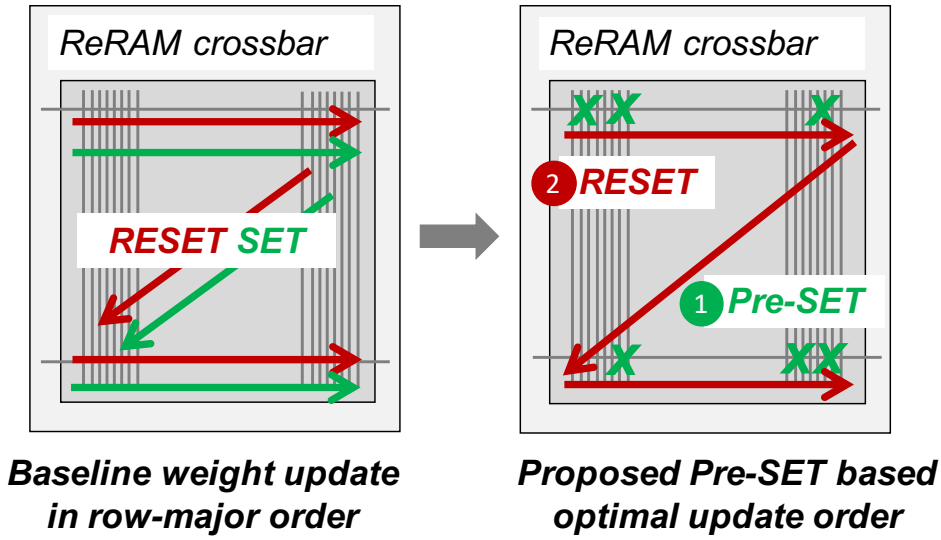


Figure 34: A comparison of the baseline weight update in row-major order and the proposed optimized programming order.

accuracy and ReRAM crossbar lifetime by tuning the programming time. Table 9 shows my study on the accuracy degradation with different switching probabilities [2]. Compared to the 8-bit fixed point weight scheme with 100% switching probability, a 95% of switching probability in LSB for both of MLP and CNN model only slightly degrades accuracy by 0.33% and 0.14% respectively. However, if a 95% of switching probability applies to both of MSB and LSB in weight matrices, training of neural network models cannot converge. Therefore, I propose to only shorten the RESET timing on LSB columns in order to avoid a significant accuracy loss.

A basic workflow of the proposed scheme is shown in Figure 35. A neural network dependent switching error tolerance, such as 5% of switching error tolerance used in this work by default, is first decided. This tolerance value is then sent to a look-up table, which maps switching probabilities to their corresponding RESET latencies, to determine the shortened RESET timing. After this, programming commands with optimized RESET timing are issued to LSB columns, while precise programming is applied to MSB columns.

[2]The experimental methodologies are presented in Section 7.4 in detail.

Table 9: Model accuracy degradation with different switching probabilities.

Weight Precision	MLP	CNN
4b-MSB (100%)-4b-LSB (100%)	97.86%	90.31%
4b-MSB (100%)-4b-LSB (95%)	97.53%	90.17%
4b-MSB (95%)-4b-LSB (95%)	27.41%	19.56%

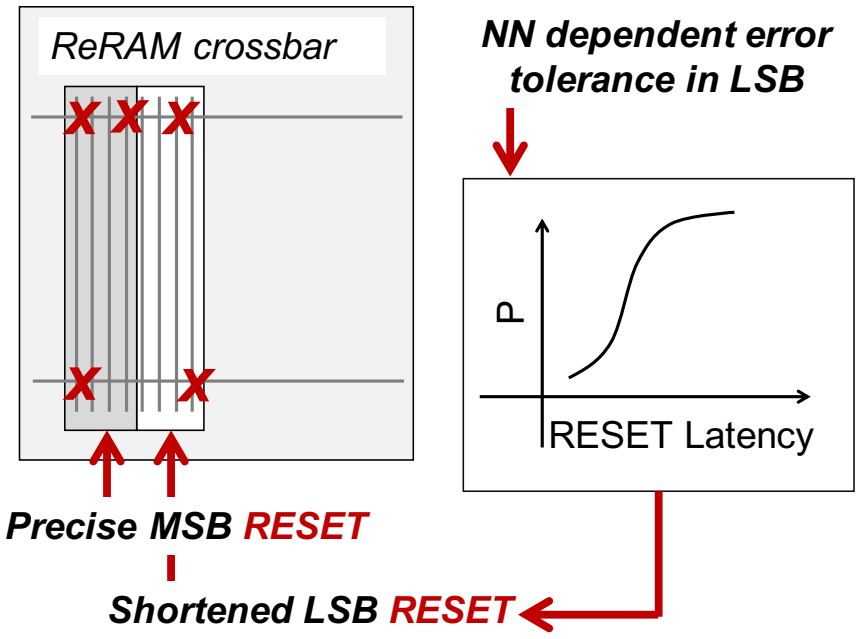


Figure 35: The precise RESET on MSB columns and shortened RESET on LSB columns.

7.3.4 Column Group Shift and Update

With shortened RESET operations on LSB columns applied, intuitively there will be a discrepancy in effective writes across LSB and MSB columns in crossbar arrays. Besides, a recent study [9] also observed a severe unbalance writes distribution in ReRAM crossbars during the neural network training. To address this issue, I propose to shift and update each half length of weight data within column groups for every training iteration, as illustrated in Figure 36. Each column group has all MSB or LSB columns from different weight data that are stored along same bitlines. With assuming 4-bit MSB/LSB and 512 rows in an array, a column group is a 512×4 data chunk that contains 512 4-bit LSB or MSB columns. This technique is inspired by a conventional wear-leveling technique for NVM based main memory [125], which is proposed to periodically shift rows and swap pages to improve endurance. However, this work proposes to shift and update weight data on all column groups in each iteration. In addition, shifting and swapping techniques for NVM based main memory incur a huge performance overhead by data exchanging, whereas the proposed shift and update scheme only requires the address remapping and it can be performed in each iteration, since weight matrices are updated in each training iteration and not necessarily preserved.

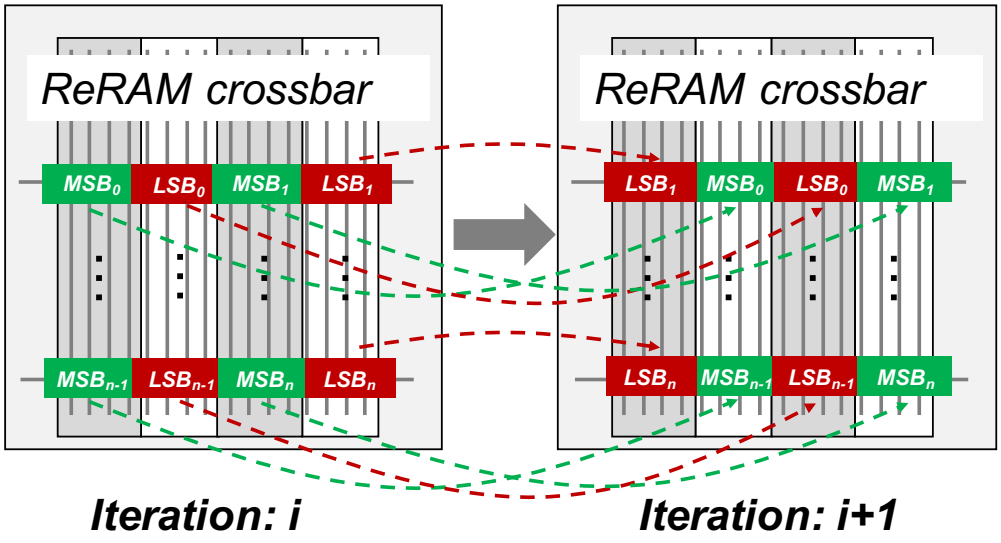


Figure 36: The proposed column group shift and update scheme.

7.4 Experimental Setup

As summarized in Table 10, I build two neural network models — MLP and CNN, with Theano [89], and then generate weight matrices by training two models on MNIST [51] and SVHN [70] datasets respectively. In this work, I test model accuracy with using different precisions of fixed-point weight matrices, and assume input data and intermediate feature maps are accurate. Batch normalization is also adopted in neural network models. For the purpose of evaluating the effectiveness of proposed techniques, I develop a simulator to estimate lifetime of ReRAM crossbars, each of which is a 512×512 array and can contain up to 64 8-bit fixed-point weights per row. This simulator takes weight matrices generated from neural network training as inputs. With a fixed-size of ReRAM crossbar, weight matrices are partitioned and reloaded onto crossbars when the size of matrix is too large for a single crossbar. In opposition, multiple small weight matrices can reside in the same physical ReRAM crossbar to maximize the utilization. The *effective write*, which is computed with Equation 4.3, is used to accumulate wearing out effect on each cell. I adopt RESET latencies from Chapter 5 with different bitline data patterns as the optimal programming strategy, and use a conservative timing for the naive solution. I also compare the proposed techniques with 512×512 MLC ReRAM crossbar baselines. I assume one MLC ReRAM cell has 16 resistance levels, i.e., 4-bit data per cell, and it is worn out 1000× (`MLC1000`) and 100× (`MLC100`) faster than SLC [72]. All evaluated schemes are summarized as below:

- `MLC100` and `MLC1000` are MLC ReRAM crossbar baselines, assuming to have 100× and 1000× effective writes than baseline SLC for each write respectively.
- `SLC` is an SLC ReRAM crossbar baseline with a conventional programming strategy.
- `SLC-OP` is an SLC ReRAM crossbar baseline with an optimized programming strategy.
- `ReNEW` has all proposed techniques to enhance lifetime of ReRAM crossbars. I evaluate `ReNEW` with different switching probabilities, `ReNEW-0.95` (95%, default), `ReNEW-0.90` (90%) and `ReNEW-0.85` (85%).

In the preceding section, I introduce my experimental methodologies. The evaluation results and analysis for the proposed techniques are presented in following sections.

Table 10: Neural networks and datasets.

Model	Dataset	Network Topology
MLP	MNIST	784-240-240-10
CNN	SVHN	conv5x32-pool3-conv4x64-pool3-1000-400-10

7.5 Accuracy and Lifetime of ReRAM Crossbars Tradeoff

Table 11 summarizes a tradeoff among the model accuracy, bit-width selection and switching probability of MLP and CNN. The baseline is a 16-bit fixed-point weight scheme, which consists of 8-bit MSB and 8-bit LSB, without switching errors (100% switching probability). In either MLP or CNN model, this baseline weight precision can achieve the best accuracy.

As reported in recent studies [126, 28, 10], using low bit-width weights for neural network training can reduce computation complexity, energy consumption while maintaining acceptable accuracy. By storing a less number of bits in weight matrices, the lifetime of ReRAM crossbars can be improved. To prove that my proposed schemes can further mitigate endurance degradation with low bit-width weight matrices, I compare 16-bit, 8-bit and 4-bit weights with no errors (100% switching probability) schemes for MLP and CNN models, and conclude that the 8-bit weight scheme for both models achieves the best tradeoff between accuracy and weight length. Using the 4-bit weight matrices in either of two models results in a failure of convergence to an acceptable accuracy.

With the 8-bit weight scheme selected, I further evaluate the impact of switching probability on accuracy. As discussed in Section 7.3, a 95% switching probability in both MSB and LSB of a weight data results in unsuccessful convergence. I then compare the accuracy with different switching probabilities, i.e., 95%, 90% and 85%, for LSB in weight data from both of MLP and CNN models. With decreased switching probabilities from 95% to 85%, the accuracy loss slightly increases in MLP model by 0.35%, 0.51% and 0.54% compared to the 16-bit

weight scheme respectively, however, the accuracy loss of CNN model dramatically increases by 0.51%, 3.23% and 34.3% respectively. Therefore, I select the 95% switching probability for the MLP model by default but a sensitivity study of different switching probabilities for LSB in weight matrices is presented in a later subsection. For the CNN model, I also select the switching probability of 95% in LSB due to an unacceptable accuracy degradation incurred by other schemes.

7.6 Lifetime Improvement

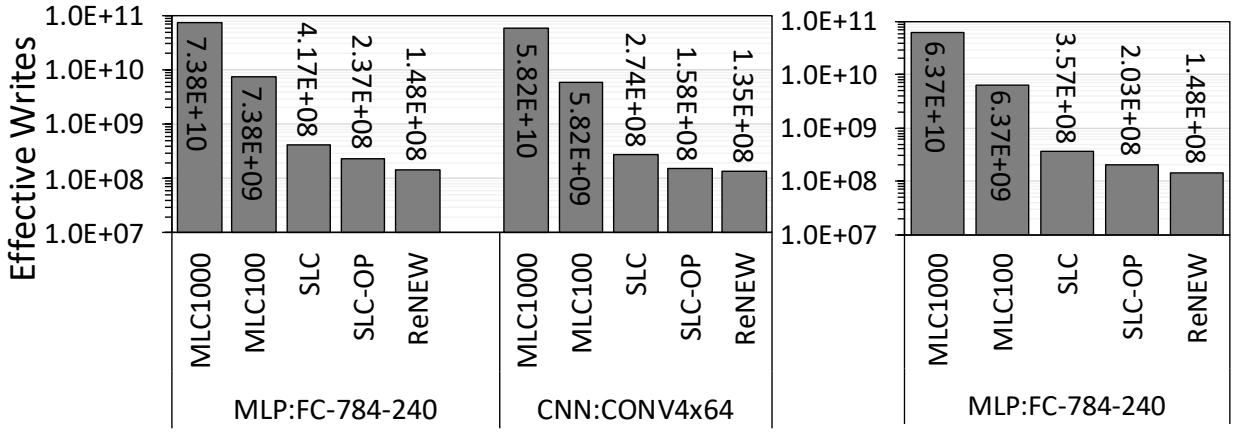
To show the effectiveness of my proposed techniques, I evaluate total effective writes, the maximum number of writes in the worst-case cell for a fully connected layer (`FC-784x240`) in the MLP model and a convolutional layer (`CONV4x64`) in the CNN model, and also conduct a sensitivity study for `ReNEW` with different switching probabilities.

7.6.1 Total Effective Writes

Figure 37 shows a comparison in total effective writes for MLP layer `FC-784x240` and CNN layer `CONV4x64`. Specifically, in Figure 37a, I compare the total effective writes among all schemes on ReRAM crossbars until their training converges to the best accuracy. For the `FC-784x240` in MLP, `ReNEW` saves 500.3×, 50.0×, 2.83× and 1.60× total effective writes compared to `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. For `CONV4x64` layer in CNN, `ReNEW` reduces 432.6×, 43.3×, 2.04× and 1.17× total effective writes compared to `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. Since training with `ReNEW` only has 84 epochs, which is 14 less than the ones with 100% switching probability, I also compare the total number of effective writes for MLP layer `FC-784x240` among all schemes with the same number of 84 training epochs (`ReNEW` and baselines for CNN layer `CONV4x64` all experience 98 epochs of training.). Figure 37b shows that, even with the same number of training epochs, `ReNEW` can still reduce total effective writes by 431.68×, 43.17×, 2.42× and 1.37× than `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively.

Table 11: Tradeoff between model accuracy loss and precisions of weight data.

MLP			**CNN**		
Weight Precision (Switching Prob.)	**Accuracy (Accuracy Loss)**	**Epoch**	**Weight Precision (Switching Prob.)**	**Accuracy (Accuracy Loss)**	**Epoch**
8b-MSB (100%)-8b-LSB (100%)	97.88% -	85	8b-MSB (100%)-8b-LSB (100%)	90.68% -	97
4b-MSB (100%)-4b-LSB (100%)	97.86% (-0.02%)	98	4b-MSB (100%)-4b-LSB 100%)	90.31% (-0.37%)	98
4b-MSB (100%)-4b-LSB (95%)	97.53% (-0.35%)	84	4b-MSB (100%)-4b-LSB (95%)	90.17% (-0.51%)	98
4b-MSB (100%)-4b-LSB (90%)	97.37% (-0.51%)	97	4b-MSB (100%)-4b-LSB (90%)	87.45% (-3.23%)	57
4b-MSB (100%)-4b-LSB (85%)	97.34% (-0.54%)	77	4b-MSB (100%)-4b-LSB (85%)	56.34% (-34.34%)	100
4b-MSB (95%)-4b-LSB (95%)	27.41% (-70.47%)	92	4b-MSB (95%)-4b-LSB (95%)	19.56% (-71.12%)	17
2b-MSB (100%)-2b-LSB (100%)	16.45% (-81.43%)	20	2b-MSB (100%)-2b-LSB (100%)	19.56% (-71.12%)	0



(a) (b)

Figure 37: Total effective writes comparison for MLP and CNN models. (a) Training with different epochs until a convergence to the best accuracy. (b) Effective writes comparison for the MLP layer `FC-784x240` among all schemes with the same number of 84 training epochs.

Additionally, I investigate and show the contribution ratio of two techniques — shortened RESET timing and optimized programming order, used to reduce total effective writes, with MLP layer `FC-784x240` in Figure 38a and CNN layer `CONV4x64` in Figure 38b. In both of two layers, the major contribution to the reduction in effective writes is from the shortened RESET timing (85.77% and 60.25% respectively). Therefore, with a greater shortened RESET timing, a larger reduction in effective writes is expected to be achieved, especially for those highly error tolerant neural networks.

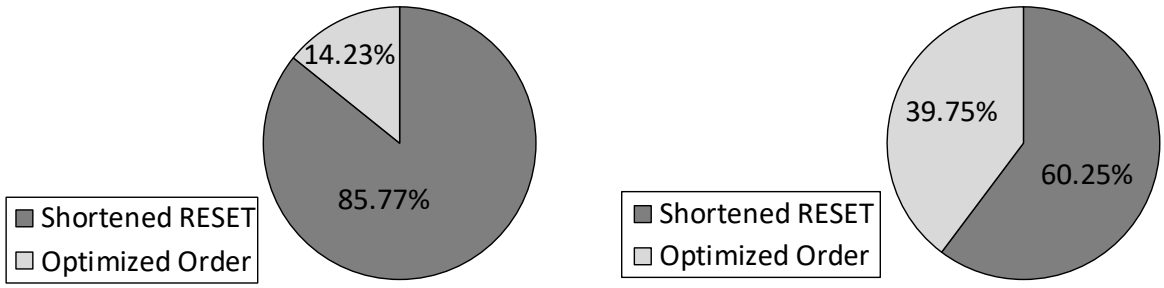
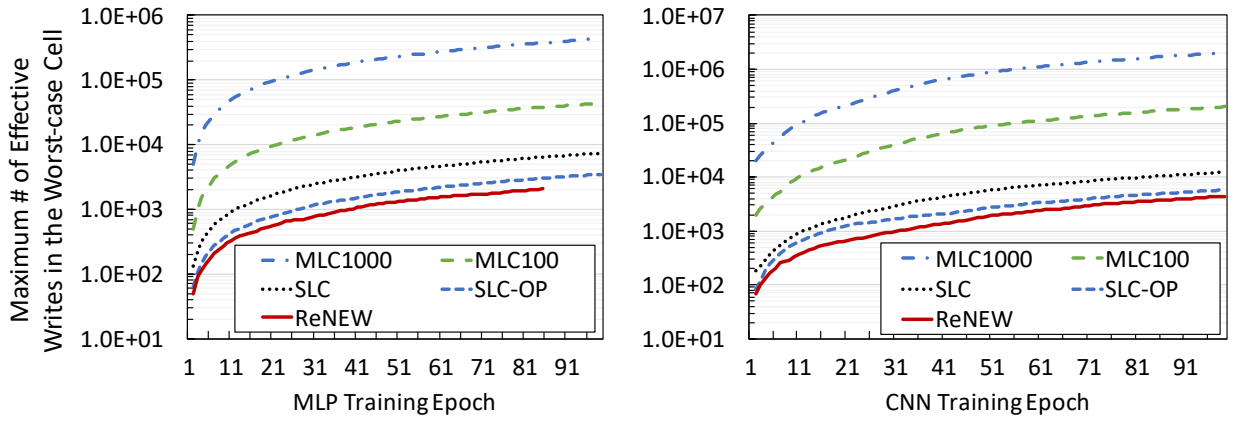


Figure 38: The contribution ratio of shortened RESET timing and optimized programming order techniques for the reduction in effective writes with (a) MLP layer `FC-784x240` and (b) CNN layer `CONV4x64`.

7.6.2 The Maximum Number of Effective Writes in Worst-case Cell

In addition to the evaluation for total effective writes, I also compare the maximum number of effective writes in the worst-case ReRAM cell, which experiences the most accumulated effective writes during the training, among all schemes with MLP layer `FC-784x240` in Figure 39a, and CNN layer `CONV4x64` in Figure 39b. For the `FC-784x240` in MLP, with 14 less epochs of training, `ReNEW` reduces the maximum number of effective writes by 212.79×, 21.28×, 3.60× and 1.70× compared to `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. However, even with the same number of training epoch of 84, `ReNEW` can still reduce the maximum effective writes by 182.67×, 18.27×, 3.13× and 1.48× than `MLC1000`, `MLC100`, `SLC` and

SLC-OP respectively. For CONV4x64 layer in CNN, ReNEW can help to improve the maximum effective writes by 460.03×, 46.00×, 2.82× and 1.33× in 98 training epochs when compared to MLC1000, MLC100, SLC and SLC-OP respectively. These experimental results prove that the proposed techniques can help to evenly distribute writes across all ReRAM cells during the neural network training.



(a) (b)

Figure 39: A comparison of the maximum number of effective writes in the worst-case ReRAM cell for (a) MLP layer FC-784x240 and (b) CNN layer CONV4x64.

7.6.3 Sensitivity to Switching Probability

As discussed previously, by decreasing the switching probability for MLP model from 95% to 85%, the accuracy loss does not significantly increase. To better understand the tradeoff between endurance improvement and switching probability, I test the lifetime enhancement for ReNEW with different switching probabilities, i.e., 95% (ReNEW-0.95), 90% (ReNEW-0.90) and 85% (ReNEW-0.85), as shown in Figure 40. Specifically, Figure 40a compares the total effective writes and accuracy loss in MLP layer FC-784x240 for ReNEW-0.95, ReNEW-0.90 and ReNEW-0.85. Overall, ReNEW-0.85 outperforms ReNEW-0.95 and ReNEW-0.90 by 1.19× and 1.31× in total effective writes reduction, while at a cost of 0.19% and 0.03% degraded accuracy respectively. ReNEW-0.90 has a larger number of total effective writes than ReNEW-0.95

since it is trained with more epochs. As shown in Figure 40b, with a same number of 77 training epochs, `ReNEW-0.90` reduces a greater number of effective writes than `ReNEW-0.95`, and `ReNEW-0.85` still has the smallest number of effective writes (1.09× and 1.04× smaller than `ReNEW-0.95` and `ReNEW-0.90` respectively). This proves that, the smaller switching probability is, a greater endurance improvement is achieved with given the same amount of updates. Figure 40c presents a comparison of maximum number of effective writes for MLP layer `FC-784x240`, wherein `ReNEW-0.85` achieves a smaller number maximum effective writes than `ReNEW-0.95` and `ReNEW-0.90` by 1.28× and 1.20× when all three schemes converge to the best accuracy. With the same number of 77 training epochs, `ReNEW-0.85` outperforms `ReNEW-0.95` and `ReNEW-0.90` by 1.10× and 1.07× respectively.

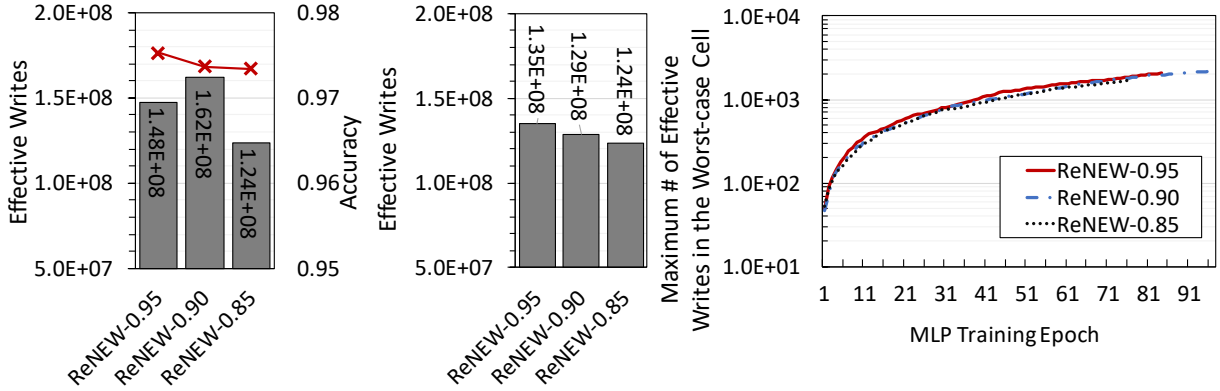


Figure 40: A sensitivity study for `ReNEW` with different switching probabilities in MLP layer `FC-784x240`: (a) total effective writes and accuracy with different training epochs, (b) total effective writes with a same number of training epochs, and (c) the maximum number of effective writes.

7.7 Conclusion

In this chapter, with the analyses of the endurance degradation mechanism in ReRAM cell, I propose a novel framework, ReNEW, to enhance the lifetime of the ReRAM crossbar based neural network accelerators, especially for neural network training in which frequent weight updates are required. The experimental evaluations show that, the proposed ReNEW reduces the total effective writes to ReRAM crossbar based accelerators by up to 500.3×, 50.0×, 2.83× and 1.60× over two MLC baselines, SLC baseline and SLC design with optimal timing, respectively.

8.0 Accelerating 3D Vertical Resistive Memory

8.1 Background and Motivation

In this section, I present a brief introduction to the 3D-VRAM array architecture and its sneak current issue.

8.1.1 3D-VRAM Array Architecture

Though 2D ReRAM crossbars have achieved a remarkable high density with a planar cell size of $4F^2$ [105] as disussed in prior chapters, to further improve the density and reduce the cost per bit for ReRAM memories remains demanding. Two typical 3D stacked ReRAM architectures, i.e., 3D-HRAM and 3D-VRAM, are proposed to overcome this challenge, both of which stem from an intuitive idea – stacking multiple layers of ReRAM crossbars vertically.

In this chapter, I focus on 3D-VRAM technology as it can be fabricated at a lower cost with a large number of vertical layers [107]. Figure 41 depicts an architecture of 3D-VRAM array. In such an architecture, multiple plane electrodes, which is connected to Word-Lines (WLs), are vertically stacked and isolated (x-y plane). The pillar electrodes are intersected perpendicularly by plane electrodes along z-axis in the figure. ReRAM cells in a 3D-VRAM are placed between plane and pillar electrodes. At the bottom of every pillar electrode, an access transistor is attached to enable addressing one single plane, which is ***essentially a planar ReRAM crossbar array (x-z plane)***, and also separates it from Bit-Lines (BLs) (along y-axis). The gates of access transistors are connected to Source-Lines (SLs) (x-axis).

8.1.2 Sneak Current Issue in 3D-VRAM Arrays

My study in the preceding chapters and many prior studies [105, 114] have demonstrated that the performance, particularly for write latency, and reliability are significantly hurt by sneak current in 2D ReRAM crossbars. Similarly, there also exists sneak current in 3D-VRAM arrays. To write or read bits from a 3D-VRAM array, WLs, SLs and BLs are

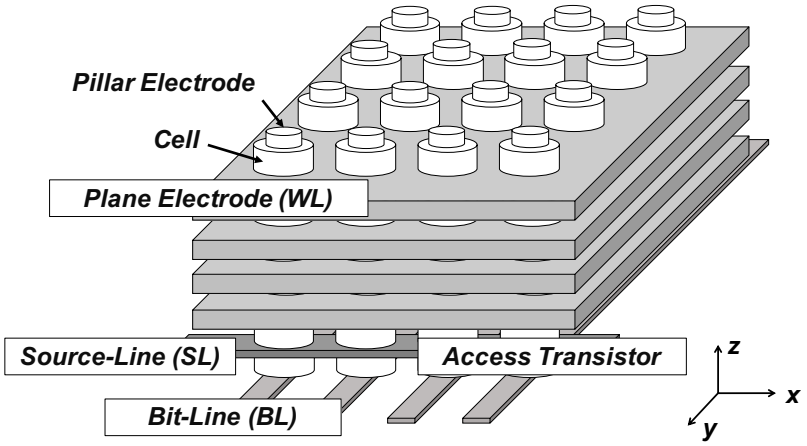


Figure 41: The architecture of a 3D-VRAM array.

properly biased with voltage supplies. For example, a write operation requires to apply V_{RESET} on the selected WL and $V_{RESET}/2$ to all other WLs, V_g on the selected SL, and $0V$ to the rest of SLs. Similarly, the selected BLs are biased to $0V$ and others are to $V_{RESET}/2$. We may notice that, except for the target cells, there a huge amount of cells are half-selected during cell switching, which thereby produce enormous sneak paths.

While an analysis for the sneak paths is critical for improving write performance in 3D-VRAM arrays, it is more challenging than that in a 2D ReRAM crossbar, since there is a larger number of half-selected cells and 3D interconnections between cells are more sophisticated. For an instance, as observed in [104], there are significantly larger number of half-selected cells on the selected WL, compared to that in 2D ReRAM crossbars. I will shortly discuss about how to mitigate the performance degradation in a 3D-VRAM array.

8.2 Proposed Designs

In this section, I elaborate my designs, including an in-memory data encoding scheme, a data pattern estimator for assessing cell resistance distributions, and a write time reduction scheme, for reducing the average write latency and, in particular, that of RESET operations.

8.2.1 Data Pattern Optimization

A key observation of voltage drop in 3D-VRAM is that the more number of LRS cells a memory array has, the larger voltage drop the target cells suffer during the RESET process. This is because larger sneak currents are generated and thereby reduce cell access voltage with more LRS cells. Consequently, limiting the number of LRS cells stored in 3D-VRAM[1] shall be able to mitigate voltage drops and improve the RESET speed. For this purpose, I propose a simple yet effective encoding scheme *Flip-n-Store*, to ensure that the number of LRS cells is no more than 50% of total cells in a 3D-VRAM array. *Flip-n-Store* focuses on minimizing the number of LRS cells in each word, as illustrated by the example as follows.

In the example shown in Figure 42, 8-bit new data that have more than $n/2$ of word length (4 in this case) LRS cells is written. Hence, flip bit is set to 1 and this new data is inverted as the new target. Though more than $n/2$-bit data are written in this case, with this data encoding scheme, there are ideally no more than 50% of cells are in LRS. By limiting the number of LRS cells, *Flip-n-Store* can also limit the number of RESET operations up to $n/2$.

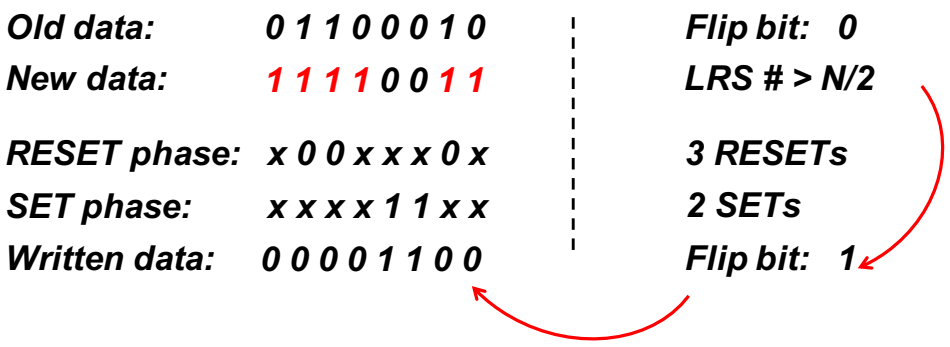


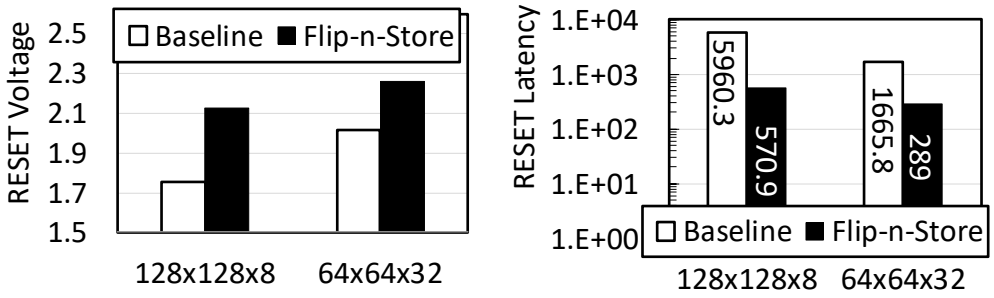
Figure 42: An example of the proposed *Flip-n-Store* scheme.

The proposed scheme *Flip-n-Store* shares similarity with previously proposed data encoding schemes for NVM [109, 17, 105]. However, they have different design goals — *Flip-n-Store* focuses on the number of LRS cells after write while previously proposed schemes are write

[1] In this work, I refer data patterns to the percentage of LRS cells in a 3D-VRAM array, and interchangeably use two terms.

optimization schemes, i.e., *DCW* [109] and *Flip-n-Write* [17] were proposed to reduce the number of modified cells at write time; [105] was proposed to reduce the number of RESET operations at write time. Therefore, as long as the number of LRS cells is reduced for the data after write, *Flip-n-Store* may increase the number of modified bits at write time.

Given that the leakage currents and voltage drops depend heavily on the number of LRS cells, *Flip-n-Store* effectively optimizes the data patterns in 3D-VRAM arrays and thus improves the write performance. To illustrate its effectiveness, I evaluate the RESET voltage stress on the target cells and RESET latency for different sizes ($N_b \times N_s \times N_l$) of 3D-VRAM arrays[2]. As shown in Figure 43, with *Flip-n-Store*, there are at most 50% of cells that are written into LRS. Consequently, the worst-case RESET voltages and latencies are significantly improved for two different 3D-VRAM array configurations.



(a) (b)

Figure 43: Comparisons of the worst-case (a) RESET voltages and (b) RESET latency between baseline and *Flip-n-Store* scheme in different sizes of 3D-VRAM arrays.

8.2.2 RESET Latency Variation

By reducing the worst-case number of LRS cells from 100% in the baseline to 50% in *Flip-n-Store*, the RESET latency can be effectively improved in 3D-VRAM. However, the 50% estimation is still pessimistic as the numbers of LRS and HRS cells are often biased,

[2]3D-VRAM modeling parameters and methodologies are presented in Section 8.3.

e.g., memory words are often initialized to 0s (HRS cells) and the high order bits of small integer values are 0s (HRS cells). As a result, there is a large gap between the performance of *Flip-n-Store* and the performance of an Oracle scheme, i.e., the hypothetical optimal scheme that knows the exact number and distribution of LRS cells.

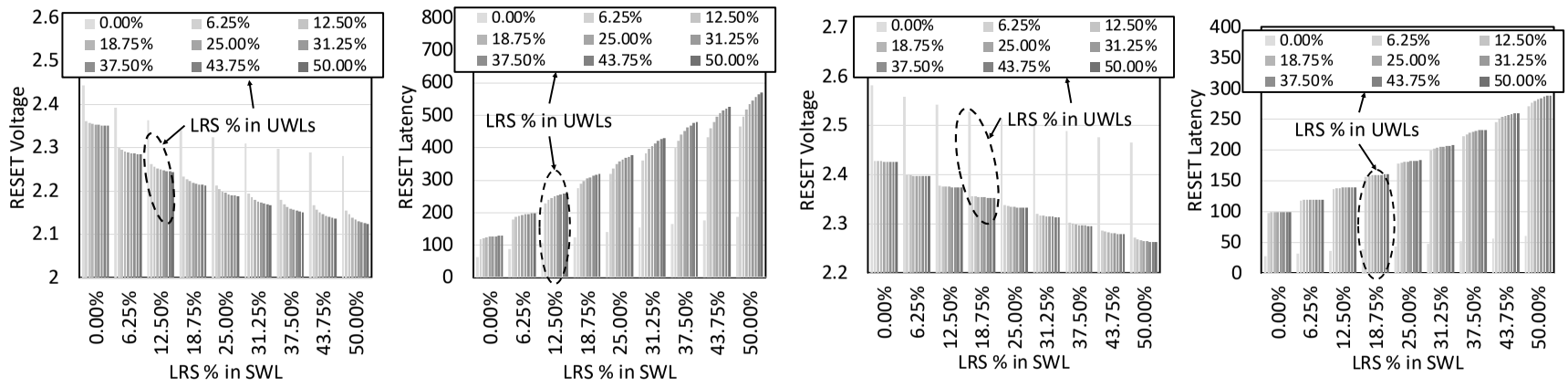
To further explore the opportunities in reducing RESET latency, I perform a study on data patterns in 3D-VRAM and summarize the results in Figure 44. The figure plots the RESET voltage drop and the RESET latency with different percentages of LRS cells in selected WLs (SWLs) and the unselected WLs (UWLs). From the figure, I observe that the percentages of LRS cells in both SWLs and UWLs are important in determining the RESET latency while the percentages of LRS cells in SWLs are more critical. When there are more LRS cells, the RESET voltages decreases (i.e., having larger voltage drops) and thus the RESET delays increase for different sizes of 3D-VRAM arrays.

While data pattern based voltage drop has been observed in 2D ReRAM arrays in Chapter 4, my study in this chapter shows that it becomes more severe in 3D-VRAM arrays. In particular, due to the large number of leakage paths in 3D-VRAM, the data patterns (i.e., the number of LRS cells) in both SWLs and UWLs determine the RESET latency while the bitline data patterns matter in 2D ReRAM crossbars. This makes it less desirable to devise a profile-guided RESET latency estimation scheme as that for 2D ReRAM arrays discussed in Chapter 5. This is because, to achieve accurate profiling, it is necessary to profile the percentages of LRS cells along different SWLs and UWLs, and create a large multi-dimensional mapping table to determine the latency at runtime. Maintaining such a table would incur large space and time overheads.

8.2.3 Data Patterns Estimation

Given that the RESET latency is highly sensitive to data patterns, I propose to leverage the analog current aggregation feature for ReRAM crossbars and adapt it in 3D-VRAM arrays for data pattern estimation.

Figure 45 illustrates how to estimate the data pattern at runtime. As shown in the figure, the percentage of LRS cells of one column (in red) is estimated using its output



(a) $128 \times 128 \times 8$ (b) $128 \times 128 \times 8$ (c) $64 \times 64 \times 32$ (d) $64 \times 64 \times 32$

Figure 44: Variations of RESET voltage and latency with LRS percentages in selected WL (SWL) and unselected WLs (UWLs) in different sizes of 3D-VRAM arrays.

analog current, which is later converted into a `dp-cnt` value. When performing the data pattern estimation, it is known that the analog current flowing through all cells that are being read is proportional to the number of LRS cells. This operation repeats in different BLs with a few rounds (which depends on the number of shared ADC units, by default, 8 ADC units are used) to complete the data pattern estimation for all cells in the WL plane. Since the estimation operation consumes energy and introduces extra latency, the `wr-cnt` is used to adjust the frequency of data profiling by only performing estimations when `wr-cnt` reaches to a predefined value. `wr-cnt` increments after a write and reset to zero after an estimation is performed.

The following summarizes the key features of the proposed pattern estimator.

- Addressing: in a 3D-VRAM array, SLs are introduced to select a certain vertical plane, and help activate one or multiple cells with BLs and WL. Therefore, in order to aggregate current flowing through all cells along the same BL, all of SLs need to be activated and turn on access transistors. A recent work [57] adopts a similar idea for neural network computing in 3D-VRAM arrays.
- Estimated data patterns: since the proposed design only cares data patterns in each vertical layer of 3D-VRAM array, a target WL plane is opened while others are grounded for the estimation purpose. The correlation between RESET latency and data patterns in 3D-VRAM arrays is a new challenge and not studied in prior work.
- Counters: the data pattern counters are attached to vertical layers, which is unlike the 2D profiler presented in Chapter 5. In each array, N_l of counters `dp-cnt` are stored for recording data patterns, and N_l of counters `wr-cnt` are used to track writes between intervals.

In the preceding section, I discuss that naively extending 2D profiler [99] to 3D-VRAM tends to incur large overhead. I next differentiate the design in Figure 45 to such a design. By extending 2D profiler in 3D-VRAM arrays, the percentages of LRS cells of SWLs and UWLs at different layers have to be profiled, recorded in different counters, and updated according when there is a write operation. A multi-dimensional lookup table to correlate the data patterns and their corresponding RESET latencies is also needed to be constructed. This

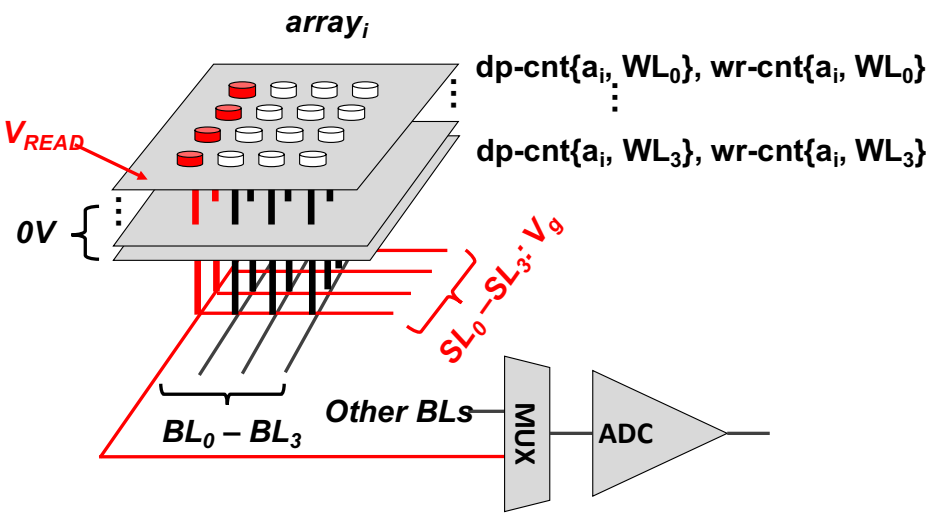


Figure 45: An example of data pattern estimation for WL planes in a 4×4×4 3D-VRAM array.

leads to unacceptable runtime and space overheads. Instead, the data pattern estimation in Figure 45 is much light-weighted. While theoretically it is less accurate, my experiments show that it achieves good tradeoff between accuracy and space/runtime overheads.

8.2.4 Write Latency Reduction with Safe and Aggressive RESET

I next elaborate the RESET latency reduction scheme with the counters for estimating the runtime data patterns. An example of the proposed RESET latency reduction scheme is presented in Figure 46. In this example, the array size is $128 \times 128 \times 8$, and the number of read/write bits n is 8 in each array, and thereby one 64B cacheline is read from or written to 64 arrays. After obtaining the data patterns, i.e., the percentage of LRS cells, an optimal RESET latency can be fetched from a lookup table based on data pattern counters. The data pattern estimation precision is 1/16, which means there are 16 equal LRS cells percentage ranges, each of which is 6.25% of total cells. The estimation frequency is defined as the number of writes that makes the data pattern counter increment at the worst-case scenario. By default, the estimation frequency is set to be 256 writes. With my data encoding scheme,

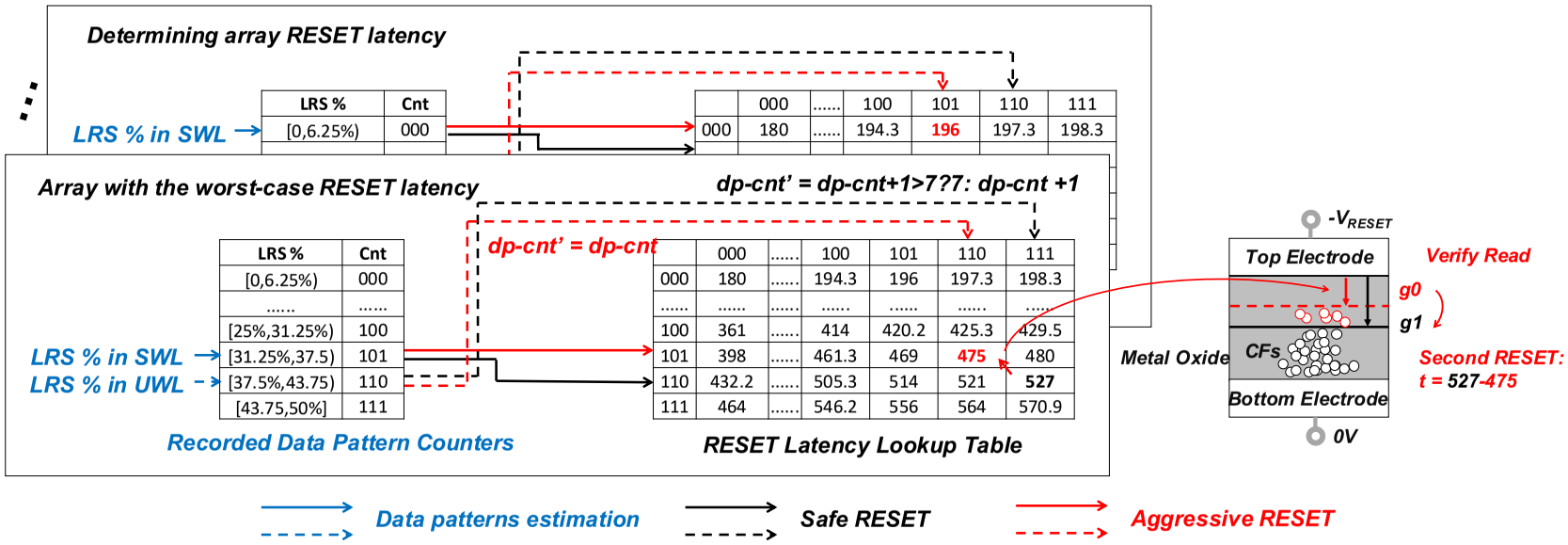


Figure 46: An example of proposed write latency reduction scheme (While a safe-RESET always finishes in one RESET round, an aggressive RESET may go through RESET-read-RESET three rounds).

each write at most brings no more than $n/2$ LRS cells into an array, i.e., equal or less than 4 LRS cells in this case. I next discuss how to determine RESET latency with array data patterns by following two schemes.

8.2.4.1 Safe RESET Though this work aims at reducing RESET latency, a successful switching should be of the first importance. This is guaranteed by following rules:

Rule 1: The worst-case latency across all 64 arrays that share one cacheline is always used.

Rule 2: After an updated data pattern is periodically obtained at runtime, there can be up to 1024 (256×4) of LRS cells during an interval. Therefore, the LRS cells percentage has to be conservatively estimated as if all new writes bring as many LRS cells as possible, and thereby it reaches to the next higher range if not exceeds 50%.

In the example, the RESET latency of $527ns$ is determined and applied to all 64 arrays in this way.

8.2.4.2 Aggressive RESET The scheme of safe RESET is the baseline for write latency reduction with conservative assumptions. It is possible to further aggressively reduce write latency by relaxing the guidelines in the scheme of safe RESET.

Since **Rule 2** assumes the worst-case scenario that does not happen often, it is feasible to relax this rule and RESET a cell with an aggressively reduced latency.

The example shown in Figure 46 illustrates this idea. Instead of using worst-case estimation, the number of LRS is assumed to have a small increase after many writes. The rationale behind this approach is that, I optimistically assume the numbers of LRS and HRS cells that are added during an interval are roughly the same. After an aggressive RESET completes, an extra read is introduced to verify if the switching is successful:

1. If it is successful, RESET is speeded up by using a shorter latency, i.e., 475ns instead of 527ns in the example. Assume the read latency is t_{READ}, the gain is (527-475-t_{READ}) ns.
2. If it is not successful, a second round RESET operation has to be issued to complete the switching. In Figure 46, if the first round RESET is not successful, a second round RESET with (527-475) ns is issued. In this case, writing the cell is finished in (527+t_{READ})

ns instead of the 527ns baseline. This is similar to the write-and-verify programming scheme widely applied in Multi-Level Cell (MLC) ReRAM [106], but here a second write is only occasionally applied if the first one was not successful.

8.2.5 Discussion

8.2.5.1 Other LRS Cells Reduction Schemes Other than the proposed encoding scheme, I also adopt compression and data shifting presented in Chapter 5 to reduce LRS cells. The compression also helps to mitigate the capacity loss of additional flip bits in the encoding scheme *Flip-n-Store*.

8.2.5.2 Overhead Since flip bits in *Flip-n-Store* consume $1/n$ storage overhead, a larger n helps to reduce this overhead. However, this also requires more sense amplifiers and write drivers, which have significant large area [104]. Therefore, a compression technique to mitigate this overhead, which is similarly adopted in [105].

The storage overhead of counters depends on the array size. For an instance, for a $128 \times 128 \times 8$ array, 8 `dp-cnt` and `wr-cnt` are used. With a 1/16 estimation precision and the encoding scheme applied, one `dp-cnt` is 3-bit and `wr-cnt` is 8-bit, and thus there is totally 11B of counter overhead per array.

For the overhead of data pattern estimator, the same ADC unit and peripheral circuitry setup is used as in Chapter 5 where eight 8-bit ADC units are shared in one bank.

8.3 Experimental Methodologies

8.3.1 3D-VRAM Array Modeling

I adopt the Verilog-A ReRAM cell device model from [40], and build and simulate an HSPICE circuit model for 3D-VRAM array with similar parameters the approaches presented in [107, 104, 56]. Table 12 summarizes the key parameters of the 3D-VRAM array.

Table 12: 3D-VRAM array model parameters.

Metric	Description	Values
N_b	Number of BLs	$128, 64$
N_s	Number of SLs	$128, 64$
N_l	Number of stacked layers	$8, 32$
n	Number of bits to read/write in parallel	8
T_{ox}	Switching layer thickness	$5nm$
d	Pillar electrode diameter	$20nm$
H_m/H_i	Thickness of a wordline plane/an isolation layer	$40nm/20nm$
F	Feature size	$22nm$
AR	Etching aspect ratio	$32, 64$
V_{WRITE}	Write voltage	$\pm 3.0V$
V_{READ}	Read voltage	$0.6V$
V_g	Gate voltage on selected SLs	$3.3V$
W/L	NMOS transistor size: width/length	$44nm/22nm$
-	Copper piller/plane interconnect resisitivity	$60\Omega \cdot nm$

8.3.2 Configuration and Simulation

In this chapter, I evaluate the memory latency, performance and dynamic energy for proposed designs with an in-house architectural simulator. Table 13 summarizes the simulation configuration. The default 3D-VRAM array size is $128 \times 128 \times 8$. In the memory latency evaluations, I use the metric *Reduction*:

$$Reduction = Latency_{BL}/Latency_{scheme} \tag{8.1}$$

where $Latency_{BL}$ and $Latency_{scheme}$ denote the write/read latency in baseline `BL` and one of proposed schemes respectively, to evaluate the effectiveness of proposed design in write latency reduction. In the evaluation for system performance, I adopt the metric *Instruction Per Cycle (IPC) Improvement*:

$$IPC\ Improvement = IPC_{scheme}/IPC_{BL} \tag{8.2}$$

where IPC_{BL} and IPC_{scheme} denote the IPC in baseline `BL` and evaluated schemes respectively. Similarly, a formula of $Reduction = Energy_{Scheme}/Energy_{BL}$ is used to evaluate the dynamic energy consumption in the experiments.

Table 13: Simulator configuration.

CPU	4 4GHz cores, single issue in-order
L1 I/D-cache	Private, 16KB/core, 4-way, 64B line, 2 cycles
L2 cache	Private, 1MB/core; 8-way, 64B line, 10 cycles
3D-VRAM memory	4GB, 1 channel, 4 ranks/channel, 8 1Gb chips/rank 8 banks/chip, 1024 $128 \times 128 \times 8$ 3D-VRAM arrays/bank READ: 18ns; SET: 10ns; RESET: latency depends on runtime data patterns

8.3.3 Benchmarks

As summarized in Table 14, I select ten benchmarks with a diversity of memory intensities from SPEC2006 [31] and PARSEC [5] benchmark suites, and collect their memory traces with Intel Pintool [61] as inputs for the architectural simulator.

Table 14: Description of benchmarks.

Write Intensity	Name	Benchmark Suite	WPKI	RPKI
High	milc	SPEC2006	54.55	95.45
	bodytrack	PARSEC	28.09	126.68
	lbm	SPEC2006	21.50	28.67
	mcf	SPEC2006	16.31	23.79
Medium	hmmer	SPEC2006	2.91	2.94
	leslie3d	SPEC2006	1.12	7.16
	sjeng	SPEC2006	0.29	0.37
Low	sphinx	SPEC2006	0.087	11.86
	calculix	SPEC2006	0.087	0.11
	x264	PARSEC	0.064	1.49

8.3.4 Compared Schemes

In the evaluations, I compare following schemes to show the effectiveness of the proposed design:

- `BL` is the baseline $128 \times 128 \times 8$ 3D-VRAM array design.
- `Flip-n-Store` is `BL` with proposed *Flip-n-Store* scheme.
- `SAFE_WR` is `Flip-n-Store` with proposed safe RESET scheme to achieve write latency reduction.

- `AGG_WR` is `Flip-n-Store` with proposed aggressive RESET scheme to opportunistically further improve write performance, which is ideally expected to show the best performance.

8.4 Evaluation Results and Analysis

8.4.1 Write Latency Reduction

Figure 47 shows the results of write latency reduction comparison among all schemes. `Flip-n-Store`, `SAFE_WR` and `AGG_WR` reduce write latency compared to `BL` by 10.28×, 22.05× and 25.98× respectively. In particular, `AGG_WR` outperforms `SAFE_WR` in write latency reduction by 17.8% on average, which shows that the opportunities for RESET latency reduction is well exploited by the proposed opportunistic RESET scheme.

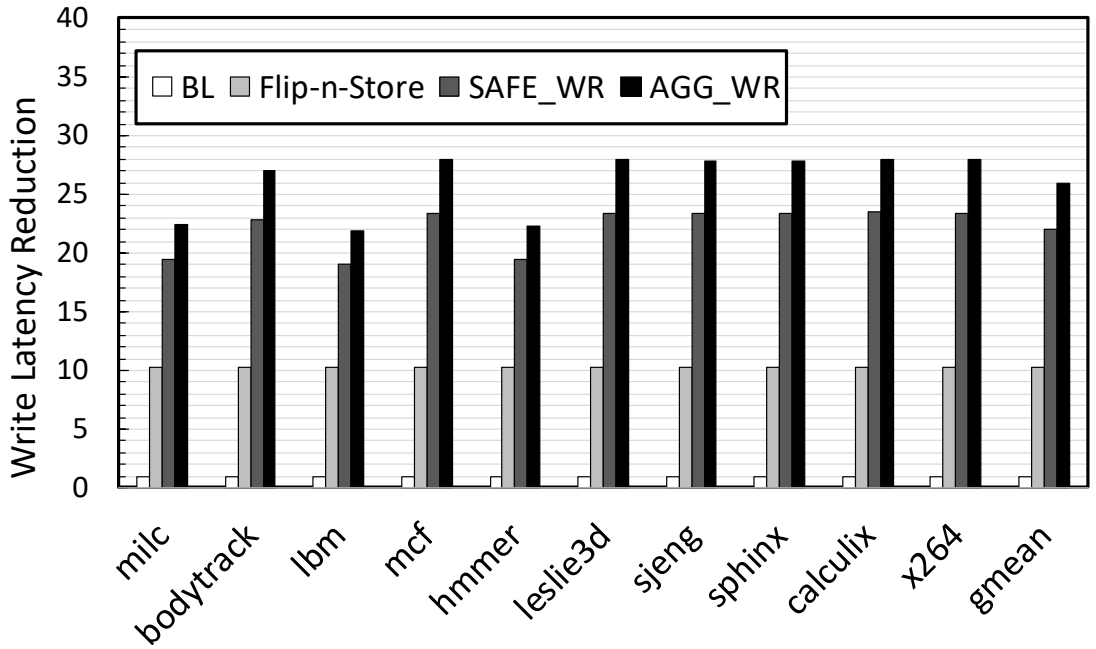


Figure 47: The write latency reduction comparison.

8.4.2 Read Latency Reduction

Figure 48 shows a comparison for memory read latency reduction across all schemes. `Flip-n-Store`, `SAFE_WR` and `AGG_WR` reduce read latency compared to `BL` by 6.91×, 11.22× and 12.32× respectively. Though my scheme focuses on write latency reduction, the average memory read latency is also effectively shortened, which is consequently helpful to improve the overall performance.

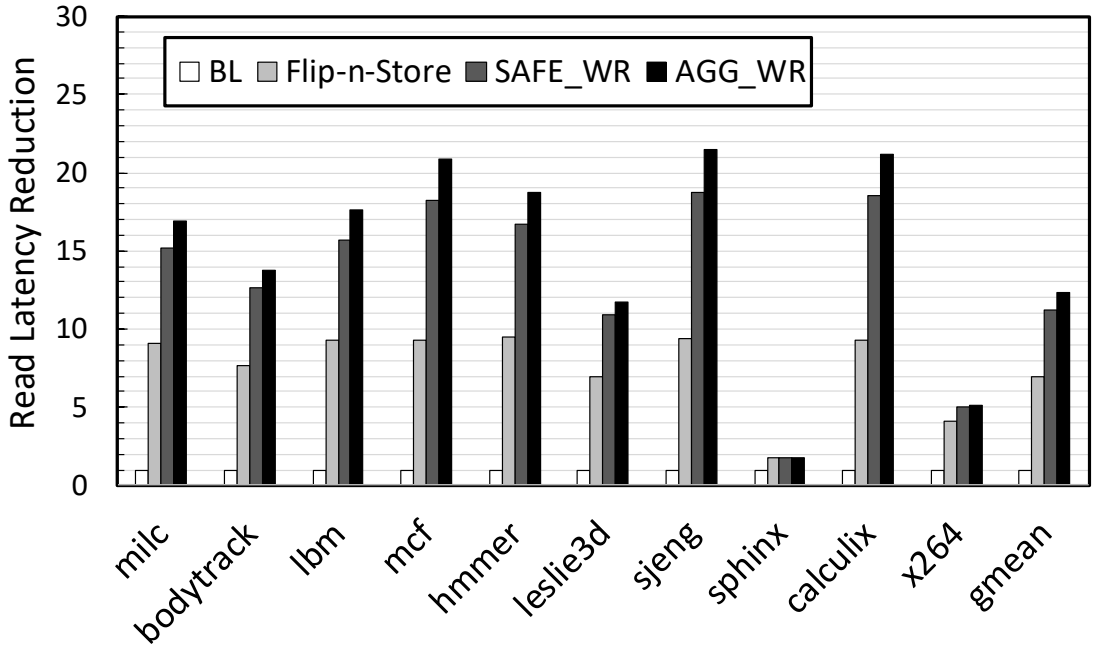


Figure 48: The read latency reduction comparison.

8.4.3 System Performance Improvement

An overall system performance comparison, which is measured by *IPC* that is normalized to `BL`, is presented in Figure 48. This figure shows that `Flip-n-Store`, `SAFE_WR` and `AGG_WR` significantly improves the overall performance compared to `BL` by 4.72×, 6.52× and 6.92× respectively. Benchmarks that have high memory intensity, in particular for write operations, has a larger improvement, since they can benefit from my proposed designs to a greater extent than those are with a lower memory intensity. Not surprisingly, `AGG_WR` outperforms

all other schemes in the system performance evaluation since it can well exploit the runtime data patterns with aggressively reducing RESET latency.

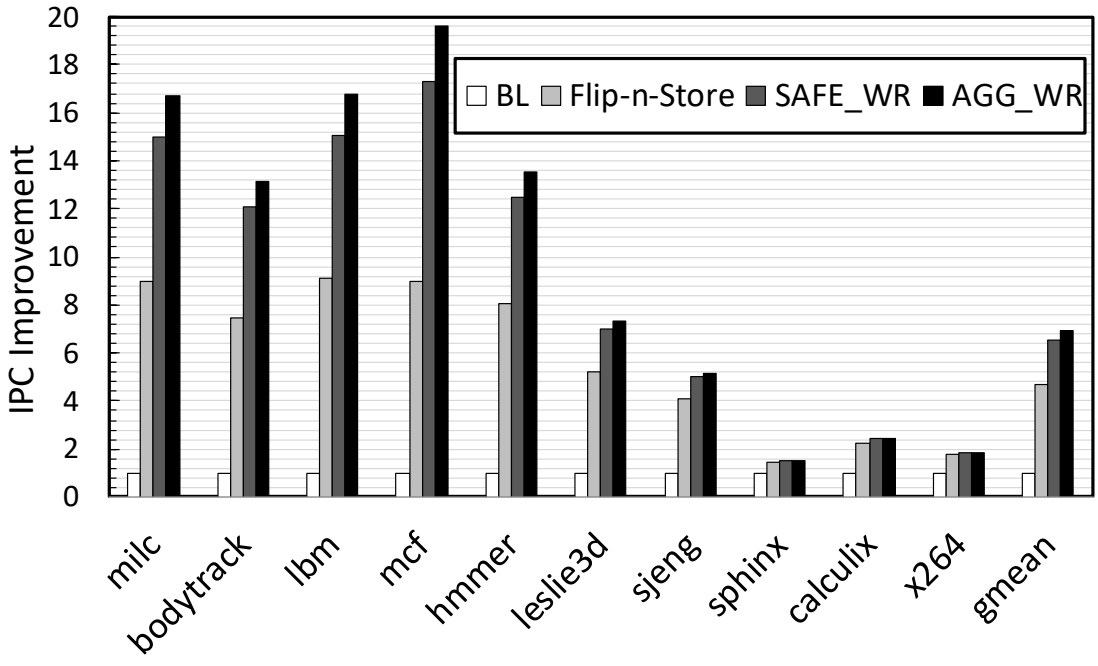


Figure 49: The IPC improvement comparison.

8.4.4 Dynamic Energy Reduction

In addition to latency and performance evaluations, a comparison of dynamic energy among all schemes is also presented in Figure 50. Dynamic memory energy primarily comes from read, write and the overhead of estimating data patterns. Memory read energy remains unchanged since the proposed design does not change read operations. The energy overhead is produced by operations of data pattern estimation, which specifically includes energy consumed by reading ReRAM cells, the ADC units and peripheral circuits used during the profiling. The evaluation shows that `Flip-n-Store`, `SAFE_WR` and `AGG_WR` greatly reduce the dynamic energy compared to `BL` by 49.8%, 52.0% and 52.4% respectively. Since the dynamic energy reduction comes from write operations, for certain benchmarks in which read operations are far more intensive than write operations, e.g., `sphinx` and `x264`, the

dynamic energy reduction is limited. Besides, on average, the energy overhead from data pattern estimations in `AGG_WR` is only 0.63%, which is negligible.

8.5 Conclusion

For the purpose of reducing write latency in 3D-VRAM arrays, in this chapter, I propose an in-memory data encoding scheme, an data pattern estimators, and an opportunistic write latency reduction scheme. Specifically, two different approaches, i.e., safe- and aggressive- RESET time estimation schemes, are presented to optimize RESET latency under the premise of successful switching, based on the runtime estimation of data patterns in a 3D-VRAM array. Experimental evaluations show that, on average, the proposed design reduces write latency by 25.98×, improves overall performance by 6.92× and reduces dynamic energy consumption by 52.4% compared to a baseline design.

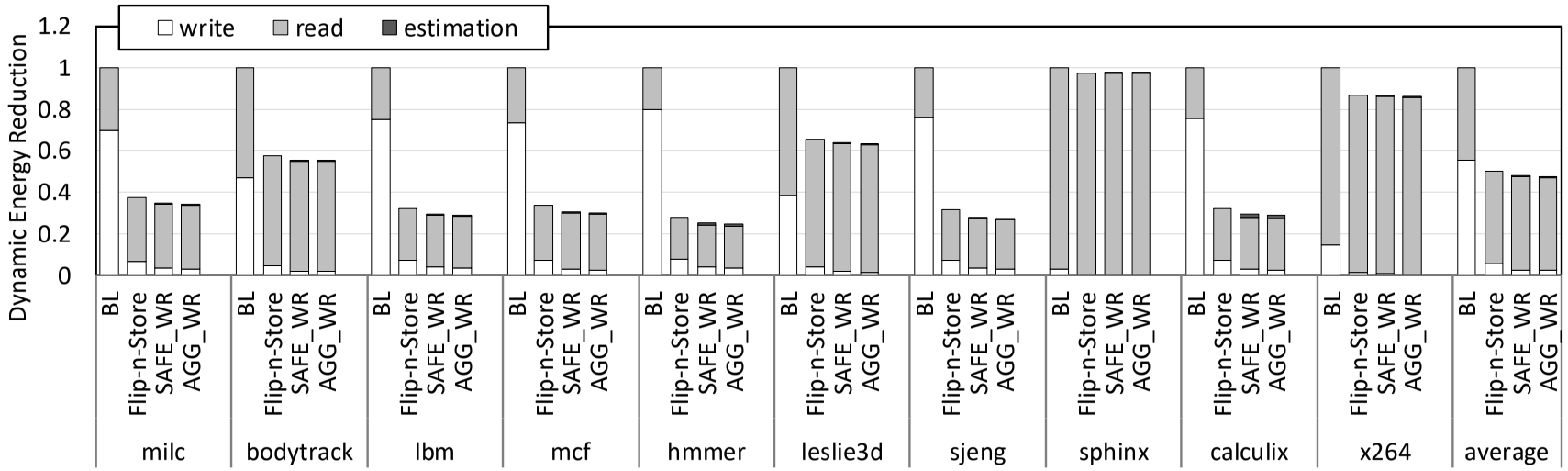


Figure 50: The dynamic energy reduction comparison.

9.0 Conclusions

This chapter concludes contributions, impacts, limitations and future research directions, of this dissertation study.

9.1 Summary of Contributions

In this dissertation, to overcome the challenges of performance and endurance issues in ReRAM crossbars, I propose a comprehensive set of techniques as follows.

I first study the correlation between the RESET latency of an ReRAM row and the number of the cells in low resistance state (LRS) on selected bitlines. I observe that the more LRS cells there are in the bitline, the larger IR drop the sneak current brings, and the longer time the RESET operation takes. Another observation is, the impact diminishes as the row becomes closer to the write driver.

Based on the observation that the RESET latency strongly correlates to the number of cells in LRS along bit lines, I propose a novel profiling-based ReRAM design, which can exploit the discrepancy of RESET latency. The in-memory processing capability of ReRAM is leveraged to implement a low overhead runtime profiler. By dynamically detecting the number of LRS cells, RESET timing is dynamically adjusted and significant performance and energy consumption improvements are archived. In addition, in order to mitigate the profiling overhead, two optimization techniques — selective profiling and fine-grained profiling, are presented. They both effectively achieve significant profiling energy reduction by reducing the number of profiling operations and halving the number of being read wordlines during a profiling operation, respectively.

For write endurance issue, I focus on mitigating the write endurance degradation from IR drop by proposing a novel wear leveling scheme for crossbar ReRAM memory. Specifically, I study the write endurance variation issue in crossbar ReRAM memory, and observe that the effective write, which indicates actual the degree of ReRAM wearing out, dynamically

changes at runtime with different data patterns and row addresses. A novel wear leveling scheme is proposed based on effective write to enhance lifetime of crossbar ReRAM memory.

Lastly, I examine two extended scenarios: (1) the endurance issue in ReRAM crossbar based accelerators for neural network computing and (2) the long write latency in 3D-VRAM arrays. For the first scenario, as such accelerators suffer from short programming cycles when weights that stored in ReRAM cells are frequently updated during the training phase, a set of schemes are proposed to prolong the lifetime of the ReRAM crossbar based accelerators, particularly for neural network training, by exploiting the wearing out mechanism of ReRAM cells. For the second scenario, an in-memory data encoding scheme to reduce the worst-case number of LRS cells, a data pattern estimator for assessing cell resistance distributions, and a write time reduction scheme that opportunistically reduces RESET latency with runtime data patterns are proposed.

9.2 Impacts

Modern applications, e.g., big data analytics, video streaming and graphical games, exhibit increasing demand for large capacity memory. However, DRAM, the *de facto* choice for main memory, faces low density, short refreshing interval and scalability challenges at 20nm and beyond [44]. Resistive Memory (ReRAM) has recently emerged as a promising candidate for architecting future large capacity main memory [101, 99, 105, 82]. It has many advantages such as non-volatility, no refreshing, high density and almost-zero standby power. Comparing to other non-volatile memory technologies, ReRAM has better density and scalability than those of STT-MRAM, and better write performance than that of PCM.

ReRAM cell arrays often adopt the crossbar architecture to achieve the smallest $4F^2$ planar cell size [105]. ReRAM crossbars enable the construction of dense main memory with large capacity, but face large sneaky currents and IR drop issues [99, 114, 82] — the leakage currents flowing through half-selected cells during writes are not negligible. Adopting access diodes helps to mitigate the issue, but cannot eliminate it completely.

To mitigate sneaky current and IR drop in ReRAM crossbars, ReRAM writes, in particular, RESET operations, conservatively adopt the worst-case latency. Recent studies have optimized the write latency from one latency fitting all cells in the crossbar to different latencies based on row address, i.e., writing the rows that are close to the write drivers can finish faster due to smaller IR drop on their cells [105, 114]. However, these prior researches did not take runtime in-memory data patterns into considerations for dynamically determining an optimal write latency. Besides, to ensure write reliability, ReRAM write drivers choose larger than ideal write voltages, which over-SET/over-RESET many cells at runtime and lead to severely degraded chip lifetime. With the issues above essentially addressed, I believe my dissertation study will benefit the academic research community and industry for the following reasons.

9.2.1 Accelerating the Deployment of Crossbar ReRAM as Main Memories

First, in the dissertation, a group of innovative solutions are proposed to overcome the challenges of performance and endurance issues in ReRAM crossbars discussed above. Since the performance and reliability of ReRAM crossbars are critical to its adoption as a replacement of DRAM, the proposed schemes enable constructing crossbar ReRAM basaed main memories with large capacity more practically in the near future, as a result of significantly reduced write latency and improved endurance. By analyzing the nonideal properties and tackling the aforementioned issues of ReRAM crossbars, this dissertation study accelerates their commercialization and deployment in modern computing systems.

9.2.2 Achieving Larger Improvements with Technology Scaling Down

Second, with technology scaling down to smaller feature sizes, we expect future ReRAM chips are fabricated with larger sizes of crossbar arrays. Whereas this fast technology advancement remarkably enlarges the capacity of ReRAM chips, the IR drop issue is expected to exacerbate due to the large wire resistance of large ReRAM crossbar arrays. Since this dissertation focuses on addressing the performance and endurance degradation caused by IR

drop issue, my proposed techniques potentially can achieve larger improvements in future ReRAM chips.

9.2.3 Highlighting the Importance of In-memory Data Patterns to Performance and Endurance

Third, based on the observations presented in Chapter 4, this dissertation reveals the importance of in-memory data patterns to performance and endurance for crossbar ReRAM. To the best of my knowledge, this work is the first architectural innovation that exploits the bitline data patterns to dynamically accelerate RESET operations in ReRAM crossbars. The proposed data pattern profiling technique creatively exploits the intrinsic analog current accumulation feature of ReRAM crossbars, while the IR drop issue itself arises from the structure and voltage biasing of crossbar arrays. I believe my dissertation study can inspire computer architects to explore innovative architectural designs for ReRAM crossbars with an emphasis on runtime data patterns and their impacts on performance and reliability.

9.2.4 Emphasizing Collaborative Efforts from Different Perspectives for Memory System Design

Fourth, the motivation of this dissertation study stems from circuit level observations, however, architectural enhancements are later proposed to mitigate the performance and endurance degradation. Therefore, this dissertation can serve as an example to show the effectiveness and significance of collaborative efforts from different perspectives, i.e., a holistic circuit-architecture study in this dissertation, to address the performance and reliability issues in modern memory systems.

9.2.5 Advancing the Development in Data Storage and Computing Applications of Crossbar ReRAM

Lastly, two extended scenarios are examined, including ReRAM crossbar based neural network accelerators and 3D-VRAM arrays, both of which are promising applications of the ReRAM crossbar structure in the near future [62, 54]. Not only for planar ReRAM crossbar based main memories, this dissertation study also proposes solutions to alleviate limited write endurance and long write latency, which are identified as major bottlenecks that prevent the development in applications of crossbar ReRAM, for computing in memory and large-capacity 3D-VRAM chips.

9.3 Limitations

The main limitations of this dissertation study come from experimental methodologies. First, due to the limited experimental capability, the evaluations for ReRAM crossbars are simulated in HSPICE with a widely used Verilog-A model [40], which only models a subset of ReRAM cells. Although this dissertation adopts a representative ReRAM cell model, however, prior studies [69, 112] revealed that the ReRAM cells with different materials can exhibit different characteristics. Besides, the evaluation results may be influenced by how accurate the analytical model in [40] reflects the switching behaviors of an ReRAM cell. Second, due to the absence of details, I can only try my best to implement the state-of-the-art design following the paper [114] and conduct a fair comparison with a same set of benchmarks in Chapter 5. However, it is worth noting that, whereas there are such limitations on ReRAM cell modeling and implementing the state-of-the-art design, the conclusion — the proposed solutions achieve significant improvements compared to either baselines or the state-of-the-art design, drawn in this dissertation remains the same. This is because: (1) the proposed techniques are expected to be applicable to a wide range of ReRAM cells and crossbars as long as they share the similar switching behaviors and the IR drop issue in general, and (2) my implementation of the state-of-the-art design has little impact on the fairness of

comparisons since the proposed design is built upon this prior work by incorporating the scheme presented in [114].

9.4 Future Research Directions

This dissertation proposes a variety of architectural solutions to improve performance and endurance for crossbar resistive memory, however, there are still potential research opportunities which are worth explorations in the future.

9.4.1 MLC ReRAM Crossbars

The first possible research direction in the future is improving performance and endurance for MLC (Multi-Level Cell) ReRAM crossbars. In this dissertation, I focus on exploring optimal solutions for SLC (Single-Level Cell) ReRAM crossbars, since SLC ReRAM crossbars are more durable and faster than its MLC counterpart. Although my proposed designs in the dissertation are also applicable to different ReRAM technologies, it is still desirable to further investigate effective solutions to improve performance and endurance for MLC ReRAM crossbars that have a much higher storage density, since they do introduce new challenges to researchers.

According to prior work [106], the write latency for MLC ReRAM is significantly longer than SLC ReRAM due to its iterative programming scheme that takes much more cycles to complete. In addition, the lifetime of an MLC ReRAM cell can be several magnitudes shorter than an SLC ReRAM cell [72]. Therefore, it is worthwhile to explore in-memory data patterns or encoding schemes in order to speed up write requests in MLC ReRAM crossbars and also improve their write endurance with architectural innovations.

9.4.2 Approximate Computing with ReRAM Crossbars

Approximate computing [118, 46, 64, 78, 42, 79] is a new computation paradigm proposed in recent years to exploit the intrinsic error-resilience feature of certain applications for

saving energy or improving performance. In an approximate computing application, instead of accurate computation results, approximate outputs with an acceptable error are sufficient. When adopting ReRAM crossbars as main memories presented in Chapter 5, 6 and 8, the successful switching is strictly guaranteed with the appropriate optimal RESET timing, while inaccurate programming is allowed to prolong the lifetime of ReRAM crossbars by exploiting wearing out mechanism and intrinsic error tolerance feature of neural network training in Chapter 7. Similar to the neural network computing, the emerging approximate computing applications also do not require error-free writes to every byte, and thereby the accurate RESET timing for programming each cell is not necessary.

As discussed in Chapter 7, the programming on ReRAM cell is a probability switching according to previous studies [55, 25, 84]. The the switching behaviors of an ReRAM cell is stochastic, and its probability is predictable by modeling the correlation between programming conditions and successful switching rate with Equation 7.3. For the nature of approximate computing applications, it is not necessary to guarantee 100% switching probability when programming ReRAM cells. Therefore, it is worthwhile to explore novel write strategies to improve performance as well as endurance for approximate computing with ReRAM crossbars.

Bibliography

[1] Armin Haj Aboutalebi and Lide Duan. Raps: Restore-aware policy selection for stt-mram-based main memory under read disturbance. In *Computer Design (ICCD), 2017 IEEE International Conference on*, pages 625–632. IEEE, 2017.

[2] Alaa Alameldeen and David Wood. Frequent pattern compression: A significance-based compression scheme for l2 caches. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2004.

[3] Kursad Albayraktaroglu, Aamer Jaleel, Xue Wu, Manoj Franklin, Bruce Jacob, Chau-Wen Tseng, and Donald Yeung. Biobench: A benchmark suite of bioinformatics applications. In *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*, pages 2–9. IEEE, 2005.

[4] Majed Valad Beigi and Gokhan Memik. Thermal-aware optimizations of reram-based neuromorphic computing systems. In *Proceedings of the 55th Annual Design Automation Conference*, page 39. ACM, 2018.

[5] Christian Bienia and Kai Li. Parsec 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, volume 2011, 2009.

[6] Mehrdad Biglari, Tobias Lieske, and Dietmar Fey. High-endurance bipolar reram-based non-volatile flip-flops with run-time tunable resistive states. In *2018 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 1–6. IEEE, 2018.

[7] Mahdi Nazm Bojnordi and Engin Ipek. Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on*, pages 1–13. IEEE, 2016.

[8] Pedro Bruel, Sai Rahul Chalamalasetti, Chris Dalton, Izzat El Hajj, Alfredo Goldman, Catherine Graves, Wen-mei Hwu, Phil Laplante, Dejan Milojicic, Geoffrey Ndu, et al. Generalize or die: Operating systems support for memristor-based accelerators. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2017.

[9] Yi Cai, Yujun Lin, Lixue Xia, Xiaoming Chen, Song Han, Yu Wang, and Huazhong Yang. Long live time: improving lifetime for training-in-memory engines by structured gradient sparsification. In *Proceedings of the 55th Annual Design Automation Conference*, page 107. ACM, 2018.

[10] Yi Cai, Tianqi Tang, Lixue Xia, Ming Cheng, Zhenhua Zhu, Yu Wang, and Huazhong Yang. Training low bitwidth convolutional neural network on rram. In *Proceedings of the 23rd Asia and South Pacific design automation conference*, pages 117–122. IEEE Press, 2018.

[11] Meng-Fan Chang, Albert Lee, Pin-Cheng Chen, Chrong Jung Lin, Ya-Chin King, Shyh-Shyuan Sheu, and Tzu-Kun Ku. Challenges and circuit techniques for energy-efficient on-chip nonvolatile memory using memristive devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):183–193, 2015.

[12] Pai-Yu Chen, Zhiwei Li, and Shimeng Yu. Design tradeoffs of vertical rram-based 3-d cross-point array. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(12):3460–3467, 2016.

[13] Pai-Yu Chen and Shimeng Yu. Compact modeling of rram devices and its applications in 1t1r and 1s1r array design. *IEEE Transactions on Electron Devices*, 62(12):4022–4028, 2015.

[14] Yang Yin Chen, Bogdan Govoreanu, Ludovic Goux, Robin Degraeve, Andrea Fantini, Gouri Sankar Kar, Dirk J Wouters, Guido Groeseneken, Jorge A Kittl, Malgorzata Jurczak, et al. Balancing set/reset pulse for $> 10^{10}$ endurance in HfO_2/Hf 1t1r bipolar rram. *IEEE Transactions on Electron devices*, 59(12):3243–3249, 2012.

[15] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE Computer Society, 2014.

[16] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *Proceedings of the 43rd International Symposium on Computer Architecture*, pages 27–39. IEEE Press, 2016.

[17] Sangyeun Cho and Hyunjin Lee. Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 347–357. IEEE, 2009.

[18] Yexin Deng, Peng Huang, Bing Chen, Xiaolin Yang, Bin Gao, Juncheng Wang, Lang Zeng, Gang Du, Jinfeng Kang, and Xiaoyan Liu. Rram crossbar array with cell selection device: A device and circuit interaction study. *IEEE Trans. Electron Devices*, 60(2):719–726, 2013.

[19] Jianbo Dong, Lei Zhang, Yinhe Han, Ying Wang, and Xiaowei Li. Wear rate leveling: Lifetime enhancement of pram with endurance variation. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 972–977. IEEE, 2011.

[20] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, July 2012.

[21] Ben Feinberg, Uday Kumar Reddy Vengalam, Nathan Whitehair, Shibo Wang, and Engin Ipek. Enabling scientific computing on memristive accelerators. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 367–382. IEEE, 2018.

[22] Ben Feinberg, Shibo Wang, and Engin Ipek. Making memristive neural network accelerators reliable. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 52–65. IEEE, 2018.

[23] Alexandre P Ferreira, Miao Zhou, Santiago Bock, Bruce Childers, Rami Melhem, and Daniel Mossé. Increasing pcm main memory lifetime. In *Proceedings of the conference on design, automation and test in Europe*, pages 914–919. European Design and Automation Association, 2010.

[24] Daichi Fujiki, Scott Mahlke, and Reetuparna Das. In-memory data parallel processor. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1–14. ACM, 2018.

[25] Siddharth Gaba, Phil Knag, Zhengya Zhang, and Wei Lu. Memristive devices for stochastic computing. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 2592–2595. IEEE, 2014.

[26] Ligang Gao, Fabien Alibart, and Dmitri B Strukov. Analog-input analog-weight dot-product operation with ag/a-si/pt memristive devices. In *VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on*, pages 88–93. IEEE, 2012.

[27] B Govoreanu, GS Kar, YY Chen, V Paraschiv, S Kubicek, A Fantini, IP Radu, L Goux, S Clima, R Degraeve, et al. 10× 10nm 2 hf/hfo x crossbar resistive ram with excellent performance, reliability and low-energy operation. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 31–6. IEEE, 2011.

[28] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[29] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. Hardware-oriented approximation of convolutional neural networks. *arXiv preprint arXiv:1604.03168*, 2016.

[30] Lei Han, Zhaoyan Shen, Zili Shao, H Howie Huang, and Tao Li. A novel reram-based processing-in-memory architecture for graph computing. In *Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2017 IEEE 6th*, pages 1–6. IEEE, 2017.

[31] John L Henning. Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, 2006.

[32] Miao Hu, John Paul Strachan, Zhiyong Li, Emmanuelle M Grafals, Noraica Davila, Catherine Graves, Sity Lam, Ning Ge, Jianhua Joshua Yang, and R Stanley Williams. Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication. In *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2016.

[33] Jiun-Jia Huang, Yi-Ming Tseng, Wun-Cheng Luo, Chung-Wei Hsu, and Tuo-Hung Hou. One selector-one resistor (1s1r) crossbar array for high-density flexible memory applications. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 31–7. IEEE, 2011.

[34] P Huang, B Chen, YJ Wang, FF Zhang, L Shen, R Liu, L Zeng, G Du, X Zhang, B Gao, et al. Analytic model of endurance degradation and its practical applications for operation scheme optimization in metal oxide based rram. In *Electron Devices Meeting (IEDM), 2013 IEEE International*, pages 22–5. IEEE, 2013.

[35] Satoru Ito, Yukio Hayakawa, Zhiqiang Wei, Shunsaku Muraoka, Koichi Kawashima, Hideto Kotani, Kazuyuki Kouno, Masayoshi Nakamura, Guo An Du, Jiann Fu Chen, et al. Reram technologies for embedded memory and further applications. In *2018 IEEE International Memory Workshop (IMW)*, pages 1–4. IEEE, 2018.

[36] Houxiang Ji, Li Jiang, Tianjian Li, Naifeng Jing, Jing Ke, and Xiaoyao Liang. Hubpa: high utilization bidirectional pipeline architecture for neuromorphic computing. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 249–254. ACM, 2019.

[37] Lei Jiang, Wujie Wen, Danghui Wang, and Lide Duan. Improving read performance of stt-mram based main memories through smash read and flexible read. In *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*, pages 31–36. IEEE, 2016.

[38] Lei Jiang, Bo Zhao, Youtao Zhang, Jun Yang, and Bruce R Childers. Improving write operations in mlc phase change memory. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–10. IEEE, 2012.

[39] Zizhen Jiang, Shengjun Qin, Haitong Li, Shosuke Fujii, Dongjin Lee, Simon Wong, and H-S Philip Wong. Selector requirements for tera-bit ultra-high-density 3d vertical rram. In *2018 IEEE Symposium on VLSI Technology*, pages 107–108. IEEE, 2018.

[40] Zizhen Jiang, Shimeng Yu, Yi Wu, Jesse H Engel, Ximeng Guan, and H-S Philip Wong. Verilog-a compact model for oxide-based resistive random access memory (rram). In *Simulation of Semiconductor Processes and Devices (SISPAD), 2014 International Conference on*, pages 41–44. IEEE, 2014.

[41] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12. IEEE, 2017.

[42] Matthias Jung, Deepak M Mathew, Christian Weis, and Norbert Wehn. Efficient reliability management in socs-an approximate dram perspective. In *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*, pages 390–394. IEEE, 2016.

[43] JF Kang, HT Li, P Huang, Z Chen, B Gao, XY Liu, ZZ Jiang, and H-SP Wong. Modeling and design optimization of reram. In *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pages 576–581. IEEE, 2015.

[44] Uksong Kang, Hak-soo Yu, Churoo Park, Hongzhong Zheng, John Halbert, Kuljit Bains, S Jang, and Joo Sun Choi. Co-architecting controllers and dram to enhance dram process scaling. In *The memory forum*, 2014.

[45] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[46] Daya S Khudia, Babak Zamirai, Mehrzad Samadi, and Scott Mahlke. Rumba: An online quality management system for approximate computing. In *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*, pages 554–566. IEEE, 2015.

[47] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[49] Lukas Kull, Thomas Toifl, Martin Schmatz, Pier Andrea Francese, Christian Menolfi, Matthias Brandli, Marcel Kossel, Thomas Morf, Toke Meyer Andersen, and Yusuf Leblebici. A 3.1 mw 8b 1.2 gs/s single-channel asynchronous sar adc with alternate comparators for enhanced speed in 32 nm digital soi cmos. *IEEE Journal of Solid-State Circuits*, 48(12):3049–3058, 2013.

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

[51] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[52] Benjamin C Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger. Phase-change technology and the future of main memory. *IEEE micro*, 30(1), 2010.

[53] Matthew Kay Fei Lee, Yingnan Cui, Thannirmalai Somu, Tao Luo, Jun Zhou, Wai Teng Tang, Weng-Fai Wong, and Rick Siow Mong Goh. A system-level simulator for rram-based neuromorphic computing chips. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(4):64, 2019.

[54] Hai Helen Li, Yiran Chen, Chenchen Liu, John Paul Strachan, and Noraica Davila. Looking ahead for resistive memory technology: A broad perspective on reram technology for future storage and computing. *IEEE Consumer Electronics Magazine*, 6(1):94–103, 2017.

[55] Haitong Li, Peng Huang, Bin Gao, Xiaoyan Liu, Jinfeng Kang, and H-S Philip Wong. Device and circuit interaction analysis of stochastic behaviors in cross-point rram arrays. *IEEE Transactions on Electron Devices*, 64(12):4928–4936, 2017.

[56] Zhiwei Li, Pai-Yu Chen, Haijun Liu, Qingjiang Li, Hui Xu, and Shimeng Yu. Quasi-analytical model of 3-d vertical-rram array architecture for mb-level design. *IEEE Transactions on Electron Devices*, 64(4):1568–1574, 2017.

[57] Zhiwei Li, Pai-Yu Chen, Hui Xu, and Shimeng Yu. Design of ternary neural network with 3-d vertical rram array. *IEEE Transactions on Electron Devices*, 64(6):2721–2727, 2017.

[58] Jiale Liang and H-S Philip Wong. Cross-point memory array without cell selectors – device characteristics and data storage pattern dependencies. *IEEE Transactions on Electron Devices*, 57(10):2531–2538, 2010.

[59] J. Lin and J. Yuan. Analysis and simulation of capacitor-less reram-based stochastic neurons for the in-memory spiking neural network. *IEEE Transactions on Biomedical Circuits and Systems*, 12(5):1004–1017, 2018.

[60] Chenchen Liu, Miao Hu, John Paul Strachan, and Hai Li. Rescuing memristor-based neuromorphic design with high defects. In *Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEE*, pages 1–6. IEEE, 2017.

[61] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *Proceedings*

of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '05, page 190–200, New York, NY, USA, 2005. Association for Computing Machinery.

[62] Siddheswar Maikap and Writam Banerjee. In quest of nonfilamentary switching: A synergistic approach of dual nanostructure engineering to improve the variability and reliability of resistive random-access-memory devices. *Advanced Electronic Materials*, page 2000209, 2020.

[63] Manqing Mao, Yu Cao, Shimeng Yu, and Chaitali Chakrabarti. Programming strategies to improve energy efficiency and reliability of reram memory systems. In *Signal Processing Systems (SiPS), 2015 IEEE Workshop on*, pages 1–6. IEEE, 2015.

[64] Joshua San Miguel, Jorge Albericio, Andreas Moshovos, and Natalie Enright Jerger. Doppelgänger: a cache for approximate computing. In *Proceedings of the 48th International Symposium on Microarchitecture*, pages 50–61. ACM, 2015.

[65] Sparsh Mittal, Jeffrey S Vetter, and Lei Jiang. Addressing read-disturbance issue in stt-ram by data compression and selective duplication. *IEEE Computer Architecture Letters*, 16(2):94–98, 2017.

[66] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P Jouppi. Cacti 6.0: A tool to model large caches. *HP laboratories*, pages 22–31, 2009.

[67] Jakob Mustafa and Rainer Waser. A novel reference scheme for reading passive resistive crossbar memories. *IEEE Transactions on Nanotechnology*, 5(6):687–691, 2006.

[68] Anirban Nag, Rajeev Balasubramonian, Vivek Srikumar, Ross Walker, Ali Shafiee, John Paul Strachan, and Naveen Muralimanohar. Newton: Gravitating towards the physical limits of crossbar acceleration. *IEEE Micro*, 38(5):41–49, 2018.

[69] C Nail, G Molas, P Blaise, G Piccolboni, B Sklenard, C Cagli, M Bernard, A Roule, M Azzaz, E Vianello, et al. Understanding rram endurance, retention and window margin trade-off using experimental results and simulations. In *Electron Devices Meeting (IEDM), 2016 IEEE International*, pages 4–5. IEEE, 2016.

[70] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[71] Leibin Ni, Zichuan Liu, Wenhao Song, J Joshua Yang, Hao Yu, Kanwen Wang, and Yuangang Wang. An energy-efficient and high-throughput bitwise cnn on sneak-path-free digital reram crossbar. In *Low Power Electronics and Design (ISLPED, 2017 IEEE/ACM International Symposium on*, pages 1–6. IEEE, 2017.

[72] Dimin Niu, Qiaosha Zou, Cong Xu, and Yuan Xie. Low power multi-level-cell resistive memory design with incomplete data mapping. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 131–137. IEEE, 2013.

[73] Ximing Qiao, Xiong Cao, Huanrui Yang, Linghao Song, and Hai Li. Atomlayer: a universal reram-based cnn accelerator with atomic layer computation. In *Proceedings of the 55th Annual Design Automation Conference*, page 103. ACM, 2018.

[74] Moinuddin K Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 14–23. ACM, 2009.

[75] Moinuddin K Qureshi, Vijayalakshmi Srinivasan, and Jude A Rivers. Scalable high performance main memory system using phase-change memory technology. *ACM SIGARCH Computer Architecture News*, 37(3):24–33, 2009.

[76] Musfiq Rahman, Bruce R Childers, and Sangyeun Cho. Comet+: Continuous online memory testing with multi-threading extension. *IEEE Transactions on Computers*, 63(7):1668–1681, 2013.

[77] Morteza Ramezani, Nima Elyasi, Mohammad Arjomand, Mahmut T Kandemir, and Anand Sivasubramaniam. Exploring the impact of memory block permutation on performance of a crossbar reram main memory. In *Workload Characterization (IISWC), 2017 IEEE International Symposium on*, pages 167–176. IEEE, 2017.

[78] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. Approximate storage in solid-state memories. *ACM Transactions on Computer Systems (TOCS)*, 32(3):9, 2014.

[79] Joshua San Miguel, Jorge Albericio, Natalie Enright Jerger, and Aamer Jaleel. The bunker cache for spatio-value approximation. In *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pages 1–12. IEEE, 2016.

[80] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S Lee. Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. *ACM SIGARCH computer architecture news*, 38(3):383–394, 2010.

[81] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *Proceedings of the 43rd International Symposium on Computer Architecture*, pages 14–26. IEEE Press, 2016.

[82] Manjunath Shevgoor, Naveen Muralimanohar, Rajeev Balasubramonian, and Yoocharn Jeon. Improving memristor memory with sneak current sharing. In *Com-*

puter Design (ICCD), 2015 33rd IEEE International Conference on, pages 549–556. IEEE, 2015.

[83] Sangho Shin, Kyungmin Kim, and Sung-Mo Kang. Analysis of passive memristive devices array: Data-dependent statistical model and self-adaptable sense resistance for rrams. *Proceedings of the IEEE*, 100(6):2021–2032, 2012.

[84] Masataka Shirasawa, Mlandeli Ernest Dlamini, and Yoshinari Kamakura. Kinetic monte carlo simulation for switching probability of reram. In *Future of Electron Devices, Kansai (IMFEDK), 2016 IEEE International Meeting for*, pages 1–1. IEEE, 2016.

[85] Linghao Song, Xuehai Qian, Hai Li, and Yiran Chen. Pipelayer: A pipelined reram-based accelerator for deep learning. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 541–552. IEEE, 2017.

[86] Dmitri B Strukov. Endurance-write-speed tradeoffs in nonvolatile memories. *Applied Physics A*, 122(4):302, 2016.

[87] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[88] Zhensen Tang, Yao Wang, Yaqing Chi, and Liang Fang. Comprehensive sensing current analysis and its guideline for the worst-case scenario of rram read operation. *Electronics*, 7(10):224, 2018.

[89] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[90] Ad J Van de Goor and Issam BS Tlili. March tests for word-oriented memories. In *Proceedings Design, Automation and Test in Europe*, pages 501–508. IEEE, 1998.

[91] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *in Deep Learning and Unsupervised Feature Learning Workshop, NIPS*. Citeseer, 2011.

[92] Chengning Wang, Dan Feng, Jingning Liu, Wei Tong, Bing Wu, and Yang Zhang. Daws: Exploiting crossbar characteristics for improving write performance of high density resistive memory. In *Computer Design (ICCD), 2017 IEEE International Conference on*, pages 281–288. IEEE, 2017.

[93] Chengning Wang, Dan Feng, Wei Tong, Jingning Liu, Zheng Li, Jiayi Chang, Yang Zhang, Bing Wu, Jie Xu, Wei Zhao, Yilin Li, and Ruoxi Ren. Cross-point resistive

memory: Nonideal properties and solutions. *ACM Trans. Des. Autom. Electron. Syst.*, 24(4), June 2019.

[94] Ching-Hua Wang, Connor McClellan, Yuanyuan Shi, Xin Zheng, Victoria Chen, Mario Lanza, Eric Pop, and H-S Philip Wong. 3d monolithic stacked 1t1r cells using monolayer mos 2 fet and hbn rram fabricated at low (150°c) temperature. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 22–5. IEEE, 2018.

[95] Weijia Wang and Bill Lin. Trained biased number representation for reram-based neural network accelerators. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2):15, 2019.

[96] Wen Wen, Youtao Zhang, and Jun Yang. Read error resilient mlc stt-mram based last level cache. In *Computer Design (ICCD), 2017 IEEE International Conference on*, pages 455–462. IEEE, 2017.

[97] Wen Wen, Youtao Zhang, and Jun Yang. Wear leveling for crossbar resistive memory. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018. © 2018 IEEE. Reprinted, with permission, from Wen Wen, Youtao Zhang and Jun Yang, Wear Leveling for Crossbar Resistive Memory, June 2018.

[98] Wen Wen, Youtao Zhang, and Jun Yang. Renew: Enhancing lifetime for reram crossbar based neural network accelerators. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pages 487–496. IEEE, 2019. © 2019 IEEE. Reprinted, with permission, from Wen Wen, Youtao Zhang and Jun Yang, ReNEW: Enhancing Lifetime for ReRAM Crossbar Based Neural Network Accelerators, November 2019.

[99] Wen Wen, Lei Zhao, Youtao Zhang, and Jun Yang. Speeding up crossbar resistive memory by exploiting in-memory data patterns. In *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*, pages 261–267. IEEE, 2017.

[100] Wen Wen, Lei Zhao, Youtao Zhang, and Jun Yang. Exploiting in-memory data patterns for performance improvement on crossbar resistive memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019. © 2019 IEEE. Reprinted, with permission, from Wen Wen, Lei Zhao, Youtao Zhang and Jun Yang, Exploiting In-memory Data Patterns for Performance Improvement on Crossbar Resistive Memory, September 2019.

[101] H-S Philip Wong, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T Chen, and Ming-Jinn Tsai. Metal–oxide rram. *Proceedings of the IEEE*, 100(6):1951–1970, 2012.

[102] Huaqiang Wu, Xiao Hu Wang, Bin Gao, Ning Deng, Zhichao Lu, Brent Haukness, Gary Bronner, and He Qian. Resistive random access memory for future information processing system. *Proceedings of the IEEE*, 2017.

[103] Wei Wu, Huaqiang Wu, Bin Gao, Ning Deng, Shimeng Yu, and He Qian. Improving analog switching in hfo x-based resistive memory with a thermal enhanced layer. *IEEE Electron Device Letters*, 38(8):1019–1022, 2017.

[104] Cong Xu, Pai-Yu Chen, Dimin Niu, Yang Zheng, Shimeng Yu, and Yuan Xie. Architecting 3d vertical resistive memory for next-generation storage systems. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 55–62. IEEE, 2014.

[105] Cong Xu, Dimin Niu, Naveen Muralimanohar, Rajeev Balasubramonian, Tao Zhang, Shimeng Yu, and Yuan Xie. Overcoming the challenges of crossbar resistive memory architectures. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pages 476–488. IEEE, 2015.

[106] Cong Xu, Dimin Niu, Naveen Muralimanohar, Norman P Jouppi, and Yuan Xie. Understanding the trade-offs in multi-level cell reram memory design. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2013.

[107] Cong Xu, Dimin Niu, Shimeng Yu, and Yuan Xie. Modeling and design analysis of 3d vertical resistive memory—a low cost cross-point architecture. In *2014 19th Asia and South Pacific design automation conference (ASP-DAC)*, pages 825–830. IEEE, 2014.

[108] Hao Yan, Lei Jiang, Lide Duan, Wei-Ming Lin, and Eugene John. Flowpap and flowrer: Improving energy efficiency and performance for stt-mram-based handheld devices under read disturbance. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):132, 2017.

[109] Byung-Do Yang, Jae-Eun Lee, Jang-Su Kim, Junghyun Cho, Seung-Yun Lee, and Byoung-Gon Yu. A low power phase-change random access memory using a data-comparison write scheme. In *2007 IEEE International Symposium on Circuits and Systems*, pages 3014–3017. IEEE, 2007.

[110] Shimeng Yu, Ximeng Guan, and H-S Philip Wong. On the switching parameter variation of metal oxide rram—part ii: Model corroboration and device design strategy. *IEEE Transactions on Electron Devices*, 59(4):1183–1188, 2012.

[111] Jianhui Yue and Yifeng Zhu. Accelerating write by exploiting pcm asymmetries. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pages 282–293. IEEE, 2013.

[112] Furqan Zahoor, Tun Zainal Azni Zulkifli, and Farooq Ahmad Khanday. Resistive random access memory (rram): an overview of materials, switching mechanism, performance, multilevel cell (mlc) storage, modeling, and applications. *Nanoscale Research Letters*, 15:1–26, 2020.

[113] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 161–170. ACM, 2015.

[114] Hang Zhang, Nong Xiao, Fang Liu, and Zhiguang Chen. Leader: Accelerating reram-based main memory by leveraging access latency discrepancy in crossbar arrays. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, pages 756–761. IEEE, 2016.

[115] Lunkai Zhang, Brian Neely, Diana Franklin, Dmitri Strukov, Yuan Xie, and Frederic T Chong. Mellow writes: Extending lifetime in resistive memories through selective slow write backs. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 519–531. IEEE, 2016.

[116] Shuhang Zhang, Grace Li Zhang, Bing Li, Hai Helen Li, and Ulf Schlichtmann. Aging-aware lifetime enhancement for memristor-based neuromorphic computing. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1751–1756. IEEE, 2019.

[117] Xian Zhang and Guangyu Sun. Toss-up wear leveling: Protecting phase-change memories from inconsistent write patterns. In *Proceedings of the 54th Annual Design Automation Conference 2017*, page 3. ACM, 2017.

[118] Xianwei Zhang, Youtao Zhang, Bruce R Childers, and Jun Yang. Drmp: Mixed precision-aware dram for high performance approximate and precise computing. In *Parallel Architectures and Compilation Techniques (PACT), 2017 26th International Conference on*, pages 53–63. IEEE, 2017.

[119] Yang Zhang, Dan Feng, Jingning Liu, Wei Tong, Bing Wu, and Caihua Fang. A novel reram-based main memory structure for optimizing access latency and reliability. In *Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEE*, pages 1–6. IEEE, 2017.

[120] Yang Zhang, Dan Feng, Wei Tong, Yu Hua, Jingning Liu, Zhipeng Tan, Chengning Wang, Bing Wu, Zheng Li, and Gaoxiang Xu. Cacf: A novel circuit architecture co-optimization framework for improving performance, reliability and energy of reram-based main memory system. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(2):22, 2018.

[121] Lei Zhao, Lei Jiang, Youtao Zhang, Nong Xiao, and Jun Yang. Constructing fast and energy efficient 1tnr based reram crossbar memory. In *The 18th International Symposium on Quality Electronic Design (ISQED)*, 2017.

[122] Lei Zhao, Youtao Zhang, and Jun Yang. Aep: An error-bearing neural network accelerator for energy efficiency and model protection. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1047–1053. IEEE, 2017.

[123] Mengying Zhao, Lei Jiang, Youtao Zhang, and Chun Jason Xue. Slc-enabled wear leveling for mlc pcm considering process variation. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.

[124] Jiantao Zhou, Kuk-Hwan Kim, and Wei Lu. Crossbar rram arrays: Selector device requirements during read operation. *IEEE Transactions on Electron Devices*, 61(5):1369–1376, 2014.

[125] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, page 14–23, New York, NY, USA, 2009. Association for Computing Machinery.

[126] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[127] Zhenhua Zhu, Hanbo Sun, Yujun Lin, Guohao Dai, Lixue Xia, Song Han, Yu Wang, and Huazhong Yang. A configurable multi-precision cnn computing framework based on single bit rram. In *Proceedings of the 56th Annual Design Automation Conference 2019*, page 56. ACM, 2019.

[128] F. Zokaee and L. Jiang. Mitigating voltage drop in resistive memories by dynamic reset voltage regulation and partition reset. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 275–286, 2020.

[129] Farzaneh Zokaee, Mingzhe Zhang, Xiaochun Ye, Dongrui Fan, and Lei Jiang. Magma: A monolithic 3d vertical heterogeneous reram-based main memory architecture. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.