Exploiting In-Memory Data Patterns for Performance Improvement on Crossbar Resistive Memory

Wen Wen, *Student Member, IEEE*, Lei Zhao, Youtao Zhang, *Member, IEEE*, and Jun Yang, *Senior Member, IEEE*

***Abstract*—Resistive memory (ReRAM) has emerged as a promising nonvolatile memory technology that may replace a significant portion of DRAM in future computer systems. ReRAM has many advantages, such as high density, low standby power, and good scalability. When adopting crossbar architecture, ReRAM cell can achieve the smallest theoretical size in fabrication, which is ideal for constructing dense memory with large capacity. However, crossbar cell structure suffers from a variety of reliability issues, which come from large voltage drops on long wires. To ensure operation reliability, ReRAM writes conservatively use the worst-case access latency of all cells in ReRAM arrays, which leads to significant performance degradation and dynamic energy waste. In this article, we study the correlation between the ReRAM cell switching latency and the number of cells in low-resistance state (LRS) along bitlines, and propose to dynamically speed up write operations based on bitline data patterns, i.e., the number of LRS cells presented in bitlines. We leverage the intrinsic in-memory processing capability of ReRAM crossbar and propose a low-overhead runtime profiler that effectively tracks the data patterns in different bitlines. To achieve further write latency reduction, we employ data compression and row address dependent memory data layout to reduce the numbers of LRS cells on bitlines. Moreover, we further present two optimization techniques, i.e., selective profiling and fine-grained profiling, to mitigate energy overhead brought by bitline data patterns tracking. The experimental results show that, on average, our 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 design, respectively.**

***Index Terms*—Crossbar array, data pattern, resistive memory (ReRAM), write performance.**

Manuscript received February 15, 2019; revised June 6, 2019; accepted August 15, 2019. Date of publication September 11, 2019; date of current version September 18, 2020. This work was supported in part by the U.S. National Science Foundation under Grant CCF-1617071, Grant CCF-1718080, Grant CCF-1725657, and Grant CCF-1910413. The preliminary version of this article was presented at the 36th International Conference on Computer Aided Design (ICCAD) in 2017 [1]. This article was recommended by Associate Editor J. Henkel. *(Corresponding author: Wen Wen.)*

W. Wen and J. Yang are with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15261 USA (e-mail: wew55@pitt.edu; juy9@pitt.edu).

L. Zhao and Y. Zhang are with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: leizhao@cs.pitt.edu; zhangyt@cs.pitt.edu).

Digital Object Identifier 10.1109/TCAD.2019.2940685

I. 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 [2]. Recent studies have proposed to construct future large capacity main memory using emerging nonvolatile memory (NVM) technologies, e.g., phase change memory (PCM) [3]–[7], spin transfer torque magnetic RAM (STT-MRAM) [8]–[11], and resistive memory (ReRAM) [12]–[23]. These memory technologies have good scalability, high density, almost zero low leakage power as well as nonvolatility 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 [24], [25] and better density and scalability than STT-MRAM [9], [26], [27]. When adopting crossbar architecture, ReRAM can achieve the smallest $4F^2$ planar cell size [13].

However, ReRAM crossbars suffer from large sneaky currents [13], [17], [28]–[30]. 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 [31], 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.

In this article, we focus on mitigating the performance degradation from IR drop. We summarize our contributions as follows.

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

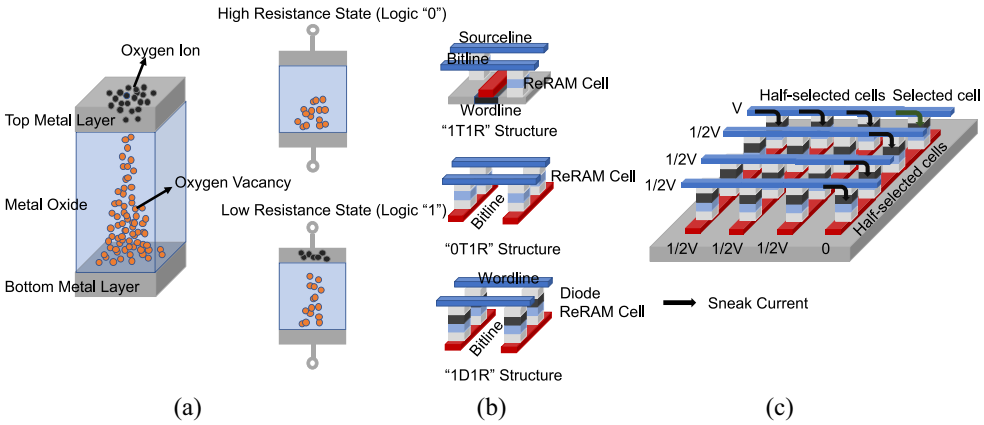


Fig. 1. ReRAM basics. (a) Cell structure. (b) Three typical ReRAM array structures. (c) Sneak current issue in ReRAM crossbar array.

to dynamically speed up the RESET operations when there are small numbers of LRS cells. We achieve further performance improvement from exploiting data compression and row address dependent data layout.

2) We 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, we periodically detect the number of LRS cells in bitlines using current aggregation, an operation having fast speed (comparable to READ operation) and low hardware and performance overheads.
3) We 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.
4) We evaluate the proposed design and compare it to the state-of-the-art. The experimental results reveal that, our 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.

In the rest of this article, we introduce the ReRAM basics and motivations in Section II. We elaborate the design details in Section III. In Section IV, we present two profiling optimization techniques, including selective profiling and fine-grained profiling. We present the experimental setup in Section V and discuss the evaluation results in Section VI. We discuss additional related work in Section VII and conclude this article in Section VIII.

II. Background and Motivation

In this section, we discuss ReRAM basics and motivate our design based on the observation of the strong correlation of the RESET latency and the number of half-selected LRS cells.

A. ReRAM Basics

ReRAM is a promising NVM technology that stores data using cell resistance. As shown in Fig. 1(a), an ReRAM cell is composed of two metal layers on the top and bottom, which are separated by metal oxide layer. Prior studies have shown that a variety of metal oxide materials, such as HfOx-based or TiOx-based materials, which have different scalability, endurance, and energy consumption characteristics, can be used to construct ReRAM cell arrays.

An ReRAM cell has two legal resistance states: an LRS to represent logic "1" and a high-resistance state (HRS) to represent logical "0." 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. The RESET operation switches the resistance state from LRS to HRS while the SET operation switches from HRS to LRS.

B. ReRAM Crossbar Structure

Fig. 1(b) 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, 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, which enables the fabrication of large cell arrays. In this article, we choose 1D1R crossbar as our baseline.

C. Motivation

We next study the sneak currents in the crossbar, and analyze its impact on ReRAM RESET latency.

For discussion purpose, we assume a cacheline has 64 B and its 512 bits are saved in 64 mats (subarrays) with each subarray containing 8 bits, the same as that in [13]. These mats spread across eight 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

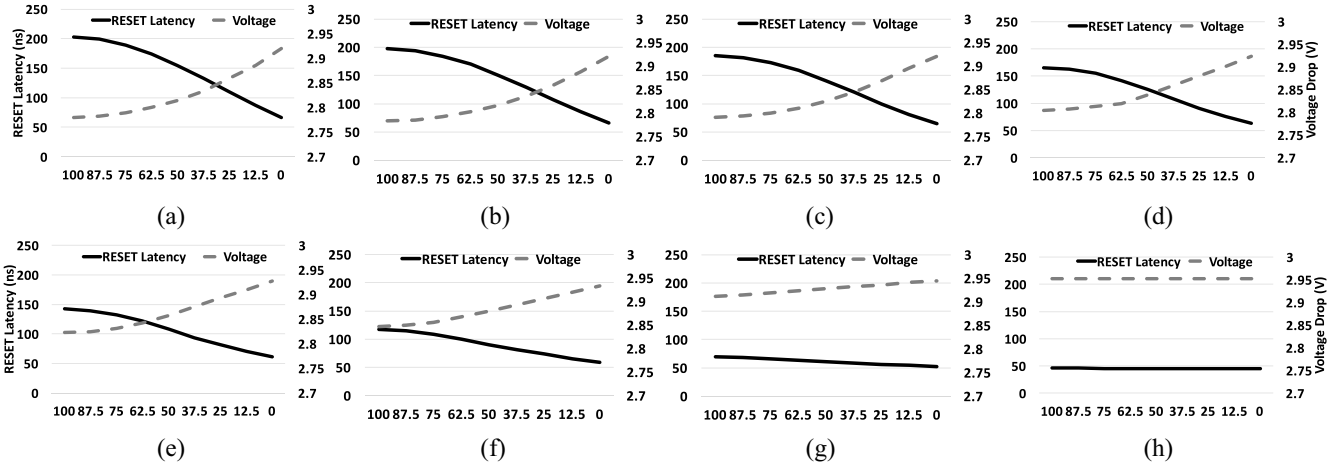


Fig. 2. (a)–(h) Variations of RESET latency and voltage drop at different LRS cell percentages in bitlines when accessing to different row address in ReRAM array. The Row Address 0 is the farthest row from driver, and Row Address 511 is the nearest row to the driver.

bitline is set to 0 V. 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 Fig. 1(c).

1) *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}.
2) *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., $V_{RESET}/2$.
3) *Not-Selected Cells:* They are the rest of the cells in the crossbar. Ideally they have no voltage stress.

A cacheline write operation consists of two phases: 1) a RESET phase to write all 0 s and 2) a SET phase to write all 1 s. We adopt *DSGB* to improve write performance [13] and *flip-n-write* to only write modified cells [32]. Based on our experiments as well as prior studies [13], [14], [17], SET operation takes much shorter time than RESET operation, making it less sensitive to voltage stress degradation. Therefore, we focus on long latency RESET operations in this article.

1) IR Drop Issue: 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 [17].

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.

2) Correlation Between RESET Latency and Number of LRS Cells: The relationship between cell RESET switching time and IR drop on the target cell can be modeled using 1, as shown in recent studies [13], [33]

$$t \times e^{kV_d} = C \tag{1}$$

where t denotes the cell RESET switching time; V_d denotes the voltage drop across the targeted cell; and C and k are the 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 0.4 V results in 10× RESET latency increase [33].

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.

Given one selected wordline and one selected bitline, we study the correlation among IR drop, the number of LRS cells, and RESET latency. Fig. 2 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] We focus on bitline LRS cells and assume the worst-case for the wordline in this article. The impact from wordline tends to be smaller due to the adoption of DSGB [13] and each subarray saving 8 bits from one cacheline. We study the RESET latency in this article, a similar observation for READ was reported in [34]. In the experiments, we adopted the Verilog-A model from [36] to build and simulate a 512 × 512 Mat circuit model in HSPICE. Table I 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

[1]Note that the term of in-memory data patterns used in this article 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 [34], [35].

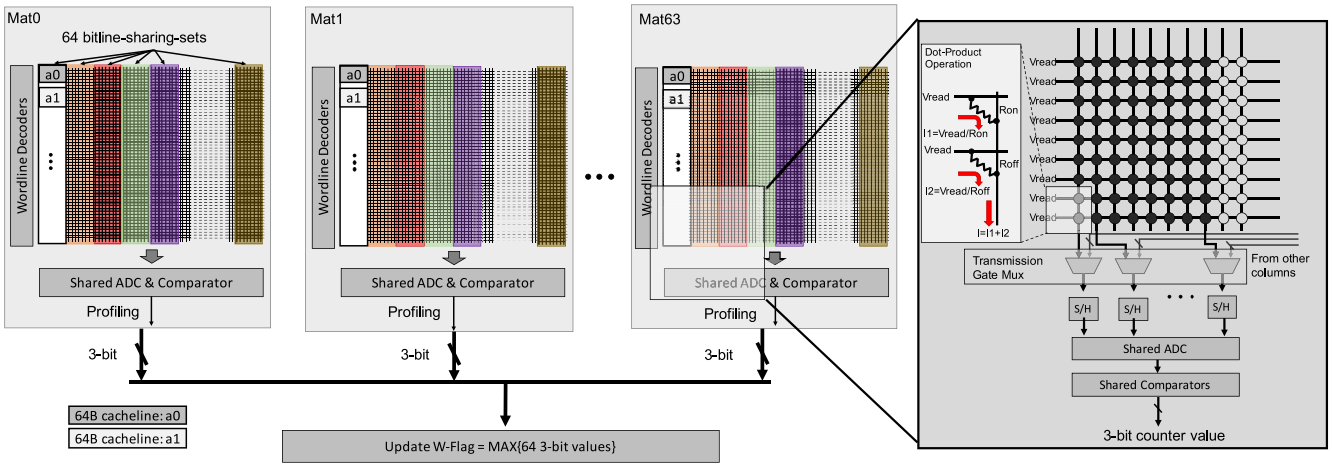


Fig. 3. Overview of the proposed architecture.

TABLE I
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

takes. 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.

Prior studies [34] 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 18 ns and 10 ns, respectively, as shown in Table IV, while RESET ranges from 56.4 ns to 202.4 ns. In this article, we 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 trade-off between limited performance improvement and increased hardware complexity.

III. DESIGN DETAILS

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

A. Overview

Fig. 3 presents an overview of our proposed scheme. We assume that each cacheline has 64 B 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 [13]. The eight 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: We attach a 3-bit flag `W-Flag` to each bitline-sharing-set. The flag records the worst-case bitline of all 512 bitlines shared by this set. In practice, we first find the worst-case bitline of each 8-bitline group in one mat, and then find the worst-case from 64 mats. 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, we divide the range [0..511] 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, we 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*.

Tracking the Worst-Case: We attach a 6-bit counter `W-Cnt` 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, we increment the counter 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. We will elaborate the use of `W-Flag` and `W-Cnt` in following sections.

RESET Latency Optimization: To RESET a memory line, we fetch its `W-Flag` and physical address to determine the appropriate tWR time for the RESET operation. By looking up a pretested RESET latency table stored in-memory controller, we can avoid always using the most conservative timing for each write. For example, if `row 0`'s `W-Flag` is 010, we may use a tWR timing of 154.6 ns

instead of 202.4 ns in the baseline design. The quantitative values of tWR timing come from our HSPICE circuit simulations, which will be discussed in a later section.

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

B. Low-Overhead Runtime Profiling

We first describe our 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 article, we leverage the current aggregation feature of ReRAM crossbar array [37], which has been widely exploited for accelerating in-memory computation [38]–[41]. Most existing memory profiling technique are for offline test. For example, March test [42] was proposed for checking memory data integrity. The test cannot be adopted at runtime as it can be as slow as 0.4 ms per row [42]–[44], which is much longer than regular ReRAM read or write operation latency.

Fig. 3 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 digital ID number (which is enough to guarantee a unique ID for each *bitline-sharing-set* in an 8 GB 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 [38].

As shown in the figure, all wordlines are applied with V_{read}; the selected eight bitlines are applied with 0 V; 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. We adopt the analog to digital conversion circuitry developed for accelerating in-memory computation. The bitline profiling currents are first sent to analogy 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 analogy to digital conversion, the largest current (corresponds to the worst-case bitline in this mat) is represented as a 3-bit digital value.

We divide the range [0..511] into eight subranges with equal size (except the last one which has one more value). As shown in Fig. 4, we set up the mapping from bitline currents to subranges before profiling. To account for runtime voltage fluctuation and cell process variations, we allocate 0.1 mA guard band for each subrange. That is, subrange "011" corresponds to LRS cell percentage range [37.5%..50%), the bitline profiling current is 1.03 mA if there are 255 LRS cells in one bitline. For high reliability, we tag a bitline as 011 as if the profiling current is 0.93 mA, 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

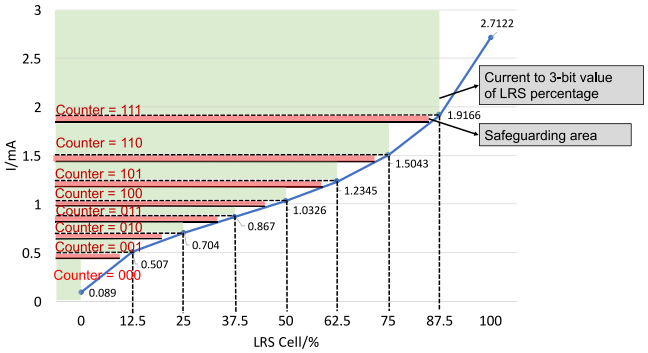


Fig. 4. Profiling current versus LRS cell percentage in 512 × 512 ReRAM crossbar array.

TABLE II
TWR (NS) FOR RESET OPERATION

LRS Ratio	Row Address Group 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

again after 64 writes so that we use 6-bit value to represent `W-Cnt`. When `W-Cnt` overflows, we may either reprofile 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.

C. Determine the RESET Timing

At runtime, we use the physical address and `W-Flag` to determine the appropriate tWR timing for the RESET operation. The reason that we also use the row address is that, similar as that in [17], 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, we split the 512 rows in one mat to eight address subranges, and use the worst-case RESET of this subgroup to write cachelines in each range, as shown in Fig. 5(a).

Table II 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 our simulations of a 512 × 512 Mat circuit model in HSPICE with parameters shown in Table I.

An Example: We next use the example in Fig. 5(b) to illustrate how our proposed online profiling works and how to determine the timing of RESET operations based on `W-Flag` and `W-Cnt`.

1) Online Profiling Operation: A profiling operation is always triggered by a `W-Cnt` overflow. The default profiling

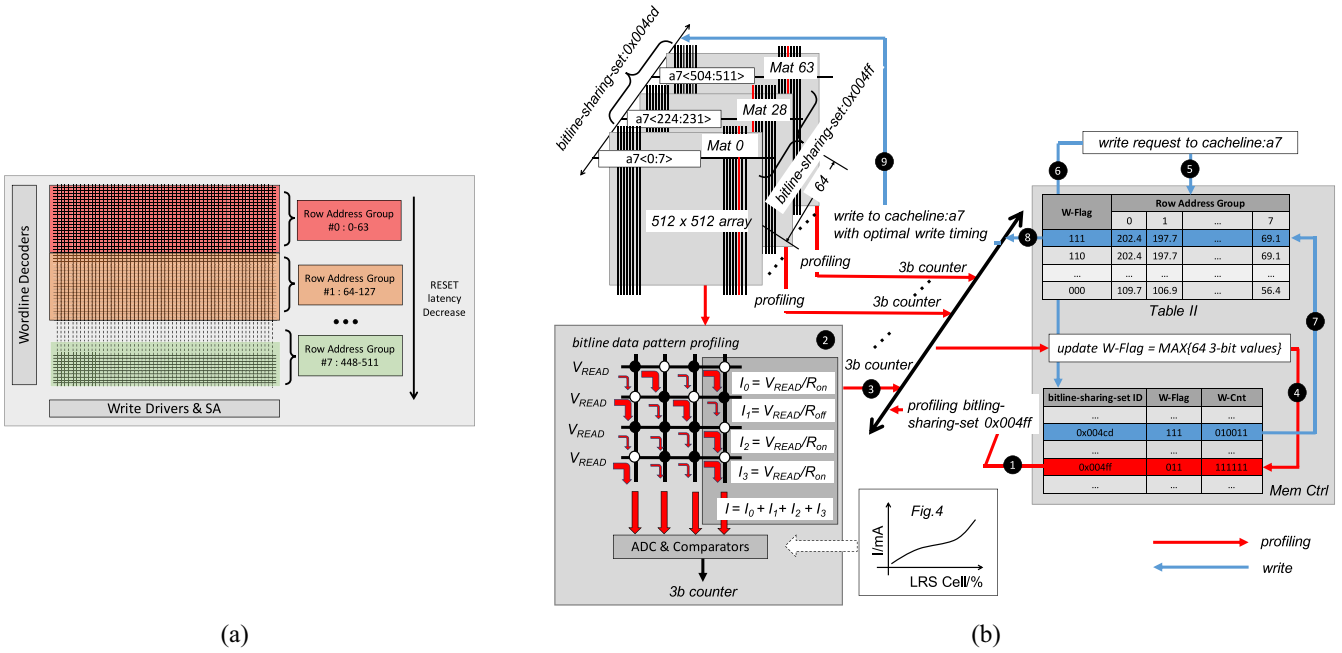


Fig. 5. (a) Rows with different addresses are mapped to eight groups with different worst-case RESET latencies. (b) Example of how our proposed online profiling works and how to determine the RESET timing.

frequency is after every 64 writes to the same 6-bit `W-Cnt` flag). For the example in Fig. 5(b), `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 an 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.

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 Fig. 5(b), a RESET operation to logic cacheline `a7` is being served. Based on its physical address, we first identify the row address group number (❺) and *bitline-sharing-set* ID (❻) (`0x004cd` in this case), and fetch an up-to-date `W-Flag` (❼). We then find the optimal RESET timing in Table II (❽) and increment `W-Cnt`. For the cells that needs to be RESET and fall in *bitline-sharing-set* `0x004cd` across 64 mats ($a7<0:7>\cdots a7<224:231>\cdots a7<504:511>$), the RESET operations can finish within the optimal RESET timing (❾).

D. 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 [46] and fill in unused cells with 0 s, i.e., reset them to HRS.

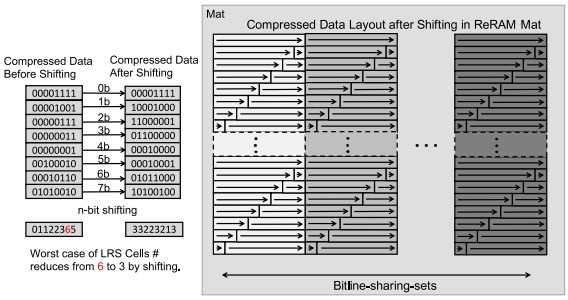


Fig. 6. Reducing LRS cells through data compress. (a) Logic view. (b) Shift in each mat.

However, we 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, we would have 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.

We 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`, we shift the row starting address to the right by `i` bits and then fill in the unused cells in the row with 0 s, as shown in Fig. 6.

E. Overhead Analysis

Profiling Overhead: The overhead comes mainly from runtime profiling. After every 64 writes to one *bitline-sharing-set*,

TABLE III
COMPARING THE PROFILING OVERHEAD IN ONE BANK

Comp.	Params	Spec	Power/Energy	Area (mm^2)
ADC [45]	sampling speed resolution number	1.28GS/s 8-bit 8	24.48mW	0.012
S+H [39]	number	8×64	5uW	0.00002
ReRAM array	Mat number Mat size	1024 512×512	Reg. Prof.: 267.178pJ Fine. Prof.: 168.332pJ Read: 72.842pJ Leakage: 255.233mW	2.078

the memory controller sends out one profiling command, which activates 64 mats. In each mat, all rows and eight bitlines are activated.

Table III summarizes the overheads for each ReRAM memory bank. We evaluated the power consumption and area by HSPICE simulation and NVSim [47] at 32 nm. A profiling operation consumes about 3.7× 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.

We followed recent studies [39], [45] to estimate the power and area overheads of adopting ADC and sampling and holding circuits. We used eight ADC units in each bank. An ADC has 1.28 GS/s sampling speed and introduces 50 ns profiling latency. In the experimental section, we 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: We attach one 3-bit `W-Flag` and 6-bit `W-Cnt` to each *bitline-sharing-set*. A *bitline-sharing-set* contains 512 64 B memory lines, or 32 KB data. For an 8-GB memory system, we need about 288 KB storage to hold all flags. In this article, we keep all flags in the memory controller for simplicity. In our future work, we 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. We expect negligible performance overhead.

IV. 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 we shall show in the experiments, the latter is much larger due to the large energy consumption from ADC units. We focus on optimizing profiling energy consumption in this section.

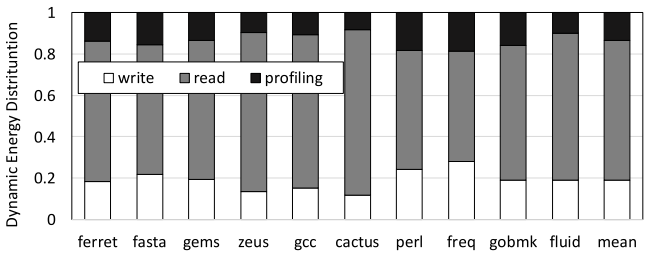


Fig. 7. Dynamic energy distribution when adopting the proposed profiling technique.

A. Profiling Energy Overhead Analysis

To better illustrate the profiling energy overhead, we compare the dynamic energy dissipation of ReRAM memory on read, write and profiling operations, respectively, for a wide range of benchmarks,[2] and summarize the results in Fig. 7. From the figure, we observe that the profiling energy consumes an average of 13.4% of total dynamic energy, a nontrivial portion of memory energy dissipation. Thus, it is important to optimize online profiling to reduce the profiling energy overhead.

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

B. Selective Profiling

Fig. 8 presents the basic idea of selective profiling. When performing the *N*th round profiling for a *bitline-sharing-set* at runtime, we find out that the worst-case LRS-cell-per-bitline number is 384 out of 512 cells, as shown by the red bar in Fig. 8(a). 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 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, we still need to perform the $N + 1$th round profiling.

To implement the proposed selective profiling scheme, we 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

[2]The experiment and simulation methodologies are discussed in Section V in detail.

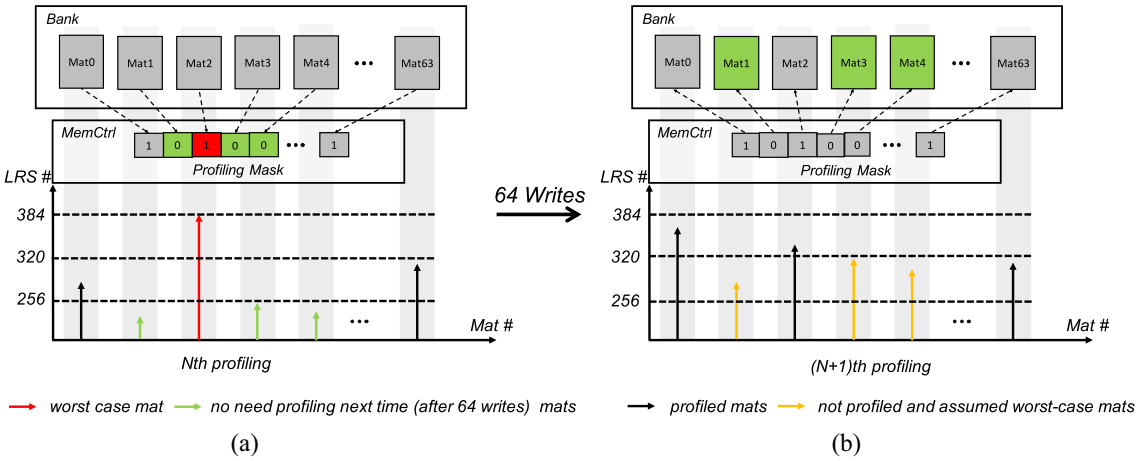


Fig. 8. Scheme of proposed selective profiling. (a) Full profiling: all mats need profiling. (b) Selective profiling: only mats in gray need profiling.

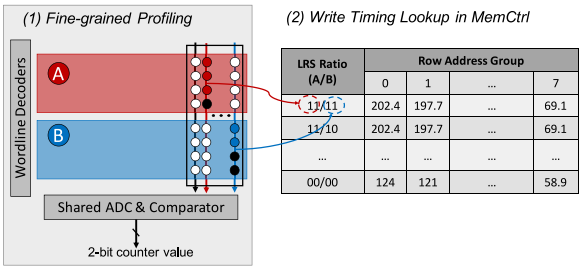


Fig. 9. Scheme of proposed fine-grained profiling.

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, we set the corresponding bit of the mat in the `profiling mask` to `0`; otherwise, the `profiling mask` bit is kept as `1`. For the next selective profiling round, we do not profile the mats whose `profiling mask` bits are 0 s.

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 twofolds: 1) it helps to save the energy consumption on the ADC/S+H circuits and the multirow read operations on ReRAM arrays and 2) it shortens the ADC latency at the sampling stage. This is because we need to process fewer samples from mats for analogy-to-digital conversion. In Section V-C, we study the performance and energy efficiency improvements in our experiments.

C. Fine-Grained Profiling

We next propose to reduce the profiling overhead as shown in Fig. 9. As aforementioned, we apply the $V_{\texttt{READ}}$ voltage 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 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. Based on this observation, we split one 512×512 ReRAM mat into two 256×512 submats. In Fig. 9, they are labeled as "A" and "B," respectively. Each submat consists of four row address groups. We profile each submat independently and use 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 to track the profiling results and to determine the RESET timing. By keeping the same profiling frequency, i.e., each submat needs to be reprofiled after accumulating 64 writes, we keep the same total number of profiling operations. 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 we profile the bitline data patterns for each submat 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 submat.

Determining the RESET timing is slightly different in the fine-grained profiling design. As shown in Fig. 9, we need to combine the two LRS ratio numbers (from submats A and B, respectively) to determine the optimal timing. Since we adopt conservative estimation, the combination may lead to overestimation, 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. This article shows that, when activating 256 wordlines during profiling, the fine-grained profiling consumes 63% energy of the one that activates all 512 wordlines (Table III).
2) Instead of having 3-bit `W-Flag` values transferred across the memory interface, we return 2-bit `W-Flag-A` and `W-Flag-B` values, which may potentially save the memory bandwidth.

TABLE IV
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 II@-3V, 88μA

TABLE V
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

3) The fine-grained profiling potentially enables the finer tuning of RESET latencies.

V. EXPERIMENTAL SETUP

In this section, we first present the modeling and simulation methodologies for evaluating the energy and performance of ReRAM crossbars. Second, we present and characterize all workloads used for evaluations based on their memory access intensity. Finally, we briefly summarize all schemes for experimental evaluations.

A. Modeling and Simulation Methodologies

To evaluate the effectiveness of our proposed design, in addition to the HSPICE modeling and simulation as introduced in Section II-C and Section III-E, we 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 IV summarizes the configuration for the baseline system. We plugged the numbers from HSPICE and NVSim [47] simulations into our architectural simulator to obtain the performance and memory energy efficiency results. We used Pintool to generate memory access traces from SPEC2006 [48], PARSEC [49], and BioBench [50] benchmark suites.

B. Workload Characterization

Table V characterizes all benchmarks used in the experiments. We carefully chose a subset of benchmarks with different memory access WPKI and RPKI in order to study the effectiveness of our design. The benchmarks are categorized to three types: high, medium, and low, respectively, according to their memory access intensity.

C. Schemes for Evaluations

In this article, we implemented and compared five different schemes, including the conventional and state-of-the-art ReRAM designs as follows.

1) `BL`: This scheme is conventional ReRAM crossbar design. The baseline adopts DSGB voltage driver for latency reduction.
2) `RA`: This scheme is the state-of-the-art design [17] that adopts row address awareness technique to reduce RESET latency.
3) `LRS`: This scheme is the naive design that only adopts data pattern profiling technique.
4) `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.
5) `PROF`: This scheme is built on top of `CMP` and includes all enhancements in this article. In particular, it adopts a 2-D tWR timing table (as shown in Table II) in determining RESET latency.

We also evaluated the effectiveness of following three schemes with profiling optimization techniques.

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

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

VI. EVALUATION RESULTS AND ANALYSIS

In this section, we 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.

A. Memory Access Latency

Fig. 10 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, we 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`.

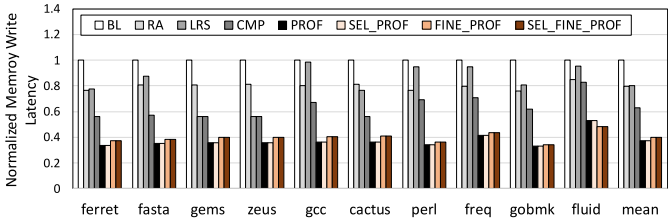


Fig. 10. Comparison of memory write latency.

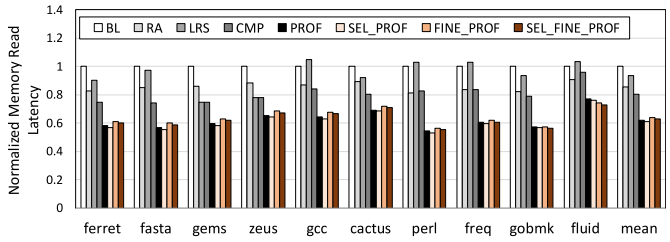


Fig. 11. Comparison of memory read latency.

The reduction of RESET latency leads to the reduction of memory read latency. Fig. 11 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. Our 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.

B. System Performance

We compared the performance when adopting different schemes and summarized the CPI (cycles-per-instruction) results in Fig. 12. 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, we also compared the proposed designs with IDEAL_PROF, the scheme adopting ideal profiling, i.e., we assume the profiling operation has zero latency and does 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.

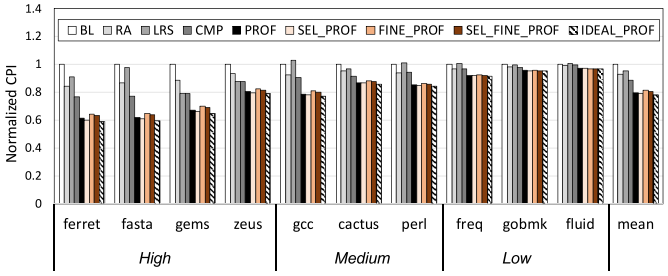


Fig. 12. Performance comparison. The benchmarks are categorized into high, medium, and low memory intensity types based on RPKI and WPKI.

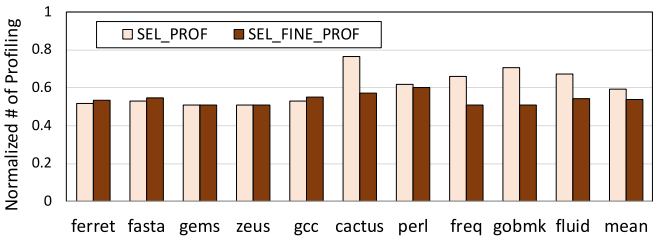


Fig. 13. Number of profiling operations performed with optimized techniques on mats (normalized to PROF).

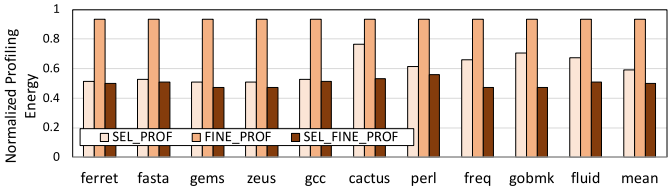


Fig. 14. Profiling energy with optimized techniques (normalized to PROF).

C. Effectiveness of Profiling Optimization

We next conducted experiments to study the effectiveness of our proposed profiling optimization techniques. We reported the normalized number of profiling operations in Fig. 13 and the normalized profiling energy consumption in Fig. 14.

Fig. 13 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%.

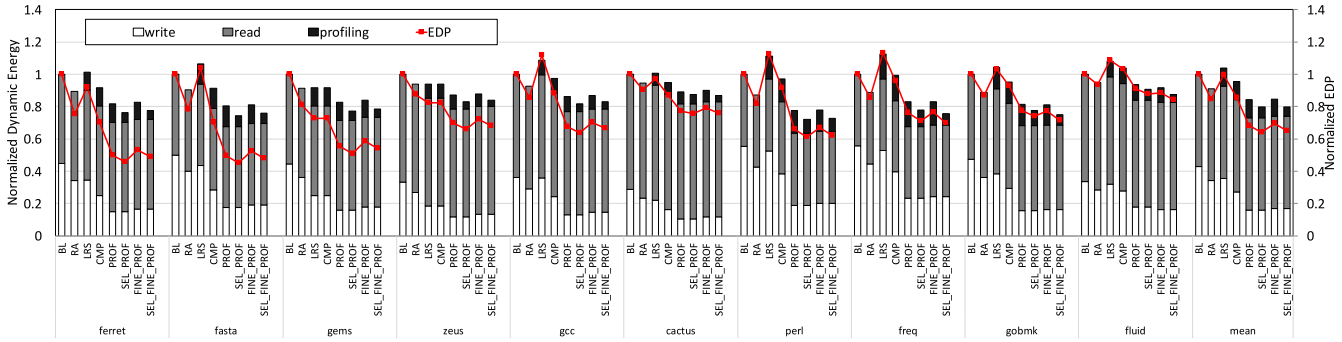


Fig. 15. Comparison of dynamic energy and EDP.

Fig. 14 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.

D. Memory Energy Efficiency

We next compared the dynamic memory energy consumption and energy-delay product (EDP) for all schemes. The results are normalized to BL and summarized in Fig. 15. The dynamic energy consumption has three major sources: read, write (including SET and RESET) energy, and profiling overheads from our proposed schemes. While the PROF greatly improves RESET performance, it has no impact on read and SET operations. In addition, our 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 our proposed design 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, FINE_PROF, and SEL_FINE_PROF also outperform RA in EDP improvements by 24.2%, 17.8%, and 23.2%, respectively.

E. Sensitivity Study

In this section, we finally compared the performance and energy efficiency results 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 Fig. 16.

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, we summarized the performance improvement and energy reduction results for scheme PROF in Fig. 16(a). From the figure, while we double the profiling area and power consumption overhead, 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.

Sensitivity to Mat Sizes: Fig. 16(b) reveals the sensitivity study results when we use different ReRAM crossbar mat sizes—we 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, we expect our proposed design can achieve larger improvements in future ReRAM arrays that have increasing mat size due to fast technology scaling.

VII. Related Work

In this section, we first introduce prior studies on improving performance of RESET operation in ReRAM crossbars, and then present related work on intrinsic current accumulation

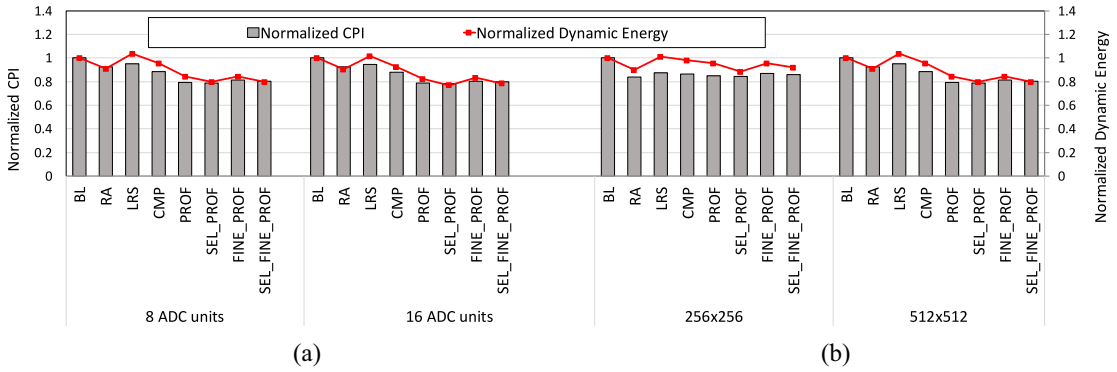


Fig. 16. Sensitivity of performance and memory dynamic energy consumption when using (a) different numbers of ADC units; and (b) different ReRAM mat sizes.

feature of ReRAM crossbars. Finally, we also report previous works on analyzing stored data pattern in ReRAM crossbars.

A. Performance of 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 [13], [14], [17], [20], [51]. Xu *et al.* [13] proposed the double sided ground biasing (DSGB), multiphase 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 the write drivers, Zhang *et al.* [17] 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.* [51] 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.* [20] proposed an ReRAM crossbar design with the double-sided write driver to reduce RESET latency.

B. Current Accumulation Feature of ReRAM Crossbars

Recent studies exploited the natural current accumulation feature of ReRAM crossbar architecture to implement dot-product analogy calculations [38]–[41], [52]–[63]. In this article, we leverage this feature to profile and track the number of LRS cells along each bitline.

C. Data Patterns in ReRAM Crossbars

Chang *et al.* [34] presented a similar observation for read operation. Mustafa and Waser [64] and Shin *et al.* [65] reported that the detection margin for read operations depends on data pattern in ReRAM arrays. Deng *et al.* [66] discussed the worst-case data patterns for read and write operations in an ReRAM crossbar array. Tang *et al.* [67] analyzed the impact of data pattern on the sensing current in ReRAM crossbars. Xu *et al.* [13] demonstrated that the RESET latency significantly increases as the number of reset bits (switched from 1 to 0) increases in an ReRAM crossbar, and then exploited the data pattern to reduce RESET latency. Liang and Wong [35] analyzed the voltage drop and data patterns in ReRAM crossbar arrays without selectors.

VIII. Conclusion

In this article, based on the observation that the RESET latency strongly correlates to the number of cells in LRSs along bit lines, we propose a novel profiling-based ReRAM design, which can exploit the discrepancy of RESET latency. We leverage the in-memory processing capability of ReRAM to implement a low-overhead runtime profiler. By dynamically detecting the number of LRS cells, we dynamically adjust RESET timing and achieve significant performance and energy consumption improvements. 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, our design improves system performance by 20.5% and 14.2%, and reduces 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, our 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.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable feedback which helped to improve the quality of this article.

References

[1] W. Wen, L. Zhao, Y. Zhang, and J. Yang, "Speeding up crossbar resistive memory by exploiting in-memory data patterns," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2017, pp. 261–267.

[2] U. Kang *et al.*, "Co-architecting controllers and DRAM to enhance DRAM process scaling," in *Proc. Memory Forum ISCA*, Jun. 2014. [Online]. Available: http://www.cs.utah.edu/thememoryforum/program14.html

[3] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," *ACM SIGARCH Comput. Architect. News*, vol. 37, no. 3, pp. 14–23, 2009.

[4] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Comput. Architect. News*, vol. 37, no. 3, pp. 24–33, 2009.

[5] B. C. Lee *et al.*, "Phase-change technology and the future of main memory," *IEEE Micro*, vol. 30, no. 1, p. 143, Jan./Feb. 2010.

[6] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proc. IEEE/ACM 42nd Annu. Int. Symp. Microarchitect.*, 2009, pp. 14–23.

[7] A. P. Ferreira, M. Zhou, S. Bock, B. R. Childers, R. G. Melhem, and D. Mossé, "Increasing PCM main memory lifetime," in *Proc. Conf. Design Autom. Test Europe*, 2010, pp. 914–919.

[8] E. Kültürsay, M. T. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2013, pp. 256–267.

[9] L. Jiang, W. Wen, D. Wang, and L. Duan, "Improving read performance of STT-MRAM based main memories through smash read and flexible read," in *Proc. IEEE 21st Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2016, pp. 31–36.

[10] H. Yan, L. Jiang, L. Duan, W.-M. Lin, and E. John, "FlowPaP and FlowReR: Improving energy efficiency and performance for STT-MRAM-based handheld devices under read disturbance," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, p. 132, 2017.

[11] A. H. Aboutalebi and L. Duan, "RAPS: Restore-aware policy selection for STT-MRAM-based main memory under read disturbance," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, 2017, pp. 625–632.

[12] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[13] C. Xu *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2015, pp. 476–488.

[14] L. Zhao, L. Jiang, Y. Zhang, N. Xiao, and J. Yang, "Constructing fast and energy efficient 1TNR based ReRAM crossbar memory," in *Proc. 18th Int. Symp. Qual. Electron. Design (ISQED)*, 2017, pp. 58–64.

[15] L. Zhang, B. Neely, D. Franklin, D. B. Strukov, Y. Xie, and F. T. Chong, "Mellow writes: Extending lifetime in resistive memories through selective slow write backs," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Architect. (ISCA)*, 2016, pp. 519–531.

[16] H. H. Li, Y. Chen, C. Liu, J. P. Strachan, and N. Davila, "Looking ahead for resistive memory technology: A broad perspective on ReRAM technology for future storage and computing," *IEEE Consum. Electron. Mag.*, vol. 6, no. 1, pp. 94–103, Jan. 2017.

[17] H. Zhang, N. Xiao, F. Liu, and Z. Chen, "Leader: Accelerating ReRAM-based main memory by leveraging access latency discrepancy in crossbar arrays," in *Proc. IEEE Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2016, pp. 756–761.

[18] S. Ito *et al.*, "ReRAM technologies for embedded memory and further applications," in *Proc. IEEE Int. Memory Workshop (IMW)*, 2018, pp. 1–4.

[19] M. Ramezani, N. Elyasi, M. Arjomand, M. T. Kandemir, and A. Sivasubramaniam, "Exploring the impact of memory block permutation on performance of a crossbar ReRAM main memory," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, 2017, pp. 167–176.

[20] Y. Zhang, D. Feng, J. Liu, W. Tong, B. Wu, and C. Fang, "A novel ReRAM-based main memory structure for optimizing access latency and reliability," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[21] W. Wen, Y. Zhang, and J. Yang, "Wear leveling for crossbar resistive memory," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, 2018, pp. 1–6.

[22] Y. Zhang *et al.*, "CACF: A novel circuit architecture co-optimization framework for improving performance, reliability and energy of ReRAM-based main memory system," *ACM Trans. Architect. Code Optim.*, vol. 15, no. 2, p. 22, 2018.

[23] M. Mao, Y. Cao, S. Yu, and C. Chakrabarti, "Programming strategies to improve energy efficiency and reliability of ReRAM memory systems," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2015, pp. 1–6.

[24] J. Yue and Y. Zhu, "Accelerating write by exploiting PCM asymmetries," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2013, pp. 282–293.

[25] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in MLC phase change memory," in *Proc. IEEE 18th Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2012, pp. 201–210.

[26] S. Mittal, J. S. Vetter, and L. Jiang, "Addressing read-disturbance issue in STT-RAM by data compression and selective duplication," *IEEE Comput. Archit. Lett.*, vol. 16, no. 2, pp. 94–98, Jul.–Dec. 2017.

[27] W. Wen, Y. Zhang, and J. Yang, "Read error resilient MLC STT-MRAM based last level cache," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, 2017, pp. 455–462.

[28] J. Zhou, K.-H. Kim, and W. Lu, "Crossbar RRAM arrays: Selector device requirements during read operation," *IEEE Trans. Electron Devices*, vol. 61, no. 5, pp. 1369–1376, May 2014.

[29] J.-J. Huang, Y.-M. Tseng, W.-C. Luo, C.-W. Hsu, and T.-H. Hou, "One selector-one resistor (1S1R) crossbar array for high-density flexible memory applications," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2011, pp. 31–37.

[30] J. F. Kang *et al.*, "Modeling and design optimization of ReRAM," in *Proc. IEEE 20th Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2015, pp. 576–581.

[31] M. Shevgoor, N. Muralimanohar, R. Balasubramonian, and Y. Jeon, "Improving memristor memory with sneak current sharing," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, 2015, pp. 549–556.

[32] S. Cho and H. Lee, "Flip-N-write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitect. (MICRO)*, 2009, pp. 347–357.

[33] B. Govoreanu *et al.*, "$10\times10\text{nm}^2$ HF/HfOx crossbar resistive RAM with excellent performance, reliability and low-energy operation," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2011, pp. 31–36.

[34] M.-F. Chang *et al.*, "Challenges and circuit techniques for energy-efficient on-chip nonvolatile memory using memristive devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 183–193, Jun. 2015.

[35] J. Liang and H.-S. P. Wong, "Cross-point memory array without cell selectors—Device characteristics and data storage pattern dependencies," *IEEE Trans. Electron Devices*, vol. 57, no. 10, pp. 2531–2538, Oct. 2010.

[36] Z. Jiang, S. Yu, Y. Wu, J. H. Engel, X. Guan, and H.-S. P. Wong, "Verilog—A compact model for oxide-based resistive random access memory (RRAM)," in *Proc. IEEE Int. Conf. Simulat. Semicond. Process. Devices (SISPAD)*, 2014, pp. 41–44.

[37] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2016, pp. 1–6.

[38] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. IEEE 43rd Int. Symp. Comput. Architect.*, 2016, pp. 27–39.

[39] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," in *Proc. 43rd Int. Symp. Comput. Architect.*, 2016, pp. 14–26.

[40] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. HPCA*, 2017, pp. 541–552.

[41] M. N. Bojnordi and E. Ipek, "Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2016, pp. 1–13.

[42] A. J. Van de Goor and I. B. S. Tlili, "March tests for word-oriented memories," in *Proc. IEEE Design Autom. Test Europe*, 1998, pp. 501–508.

[43] X. Zhang, Y. Zhang, B. R. Childers, and J. Yang, "DRMP: Mixed precision-aware DRAM for high performance approximate and precise computing," in *Proc. IEEE 26th Int. Conf. Parallel Architect. Compilation Techn. (PACT)*, 2017, pp. 53–63.

[44] M. Rahman, B. R. Childers, and S. Cho, "COMeT+: Continuous online memory testing with multi-threading extension," *IEEE Trans. Comput.*, vol. 63, no. 7, pp. 1668–1681, Jul. 2014.

[45] L. Kull *et al.*, "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 J. Solid-State Circuits*, vol. 48, no. 12, pp. 3049–3058, Dec. 2013.

[46] A. R. Alameldeen and D. A. Wood, "Frequent pattern compression: A significance-based compression scheme for L2 caches," Comput. Sci. Dept., Univ. Wisconsin–Madison, Madison, WI, USA, Rep. 1500, 2004.

[47] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[48] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Architect. News*, vol. 34, no. 4, pp. 1–17, 2006.

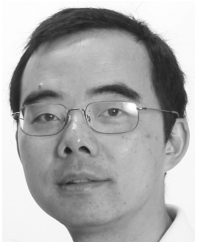
[49] C. Bienia and K. Li, "PARSEC 2.0: A new benchmark suite for chip-multiprocessors," in *Proc. 5th Annu. Workshop Model. Benchmarking Simulat.*, Jun. 2009. [Online]. Available: http://www-mount.ece.umn.edu/∼jjyi/MoBS/2009/program/02E-Bienia.pdf
[50] K. Albayraktaroglu *et al.*, "BioBench: A benchmark suite of bioinformatics applications," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2005, pp. 2–9.
[51] C. Wang, D. Feng, J. Liu, W. Tong, B. Wu, and Y. Zhang, "DAWS: Exploiting crossbar characteristics for improving write performance of high density resistive memory," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, 2017, pp. 281–288.
[52] D. Fujiki, S. A. Mahlke, and R. Das, "In-memory data parallel processor," in *Proc. ACM 23rd Int. Conf. Architect. Support Program. Lang. Oper. Syst.*, 2018, pp. 1–14.
[53] A. Nag *et al.*, "Newton: Gravitating towards the physical limits of crossbar acceleration," *IEEE Micro*, vol. 38, no. 5, pp. 41–49, Sep./Oct. 2018.
[54] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, "Enabling scientific computing on memristive accelerators," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Architect. (ISCA)*, 2018, pp. 367–382.
[55] B. Feinberg, S. Wang, and E. Ipek, "Making memristive neural network accelerators reliable," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2018, pp. 52–65.
[56] M. K. F. Lee *et al.*, "A system-level simulator for RRAM-based neuromorphic computing chips," *ACM Trans. Architect. Code Optim.*, vol. 15, no. 4, p. 64, 2019.
[57] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2017, pp. 1–6.
[58] L. Ni *et al.*, "An energy-efficient and high-throughput bitwise CNN on sneak-path-free digital ReRAM crossbar," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, 2017, pp. 1–6.
[59] M. V. Beigi and G. Memik, "Thermal-aware optimizations of ReRAM-based neuromorphic computing systems," in *Proc. 55th Annu. Design Autom. Conf.*, 2018, p. 39.
[60] H. Ji, L. Jiang, T. Li, N. Jing, J. Ke, and X. Liang, "HUBPA: High utilization bidirectional pipeline architecture for neuromorphic computing," in *Proc. ACM 24th Asia South Pac. Design Autom. Conf.*, 2019, pp. 249–254.
[61] P. Bruel *et al.*, "Generalize or die: Operating systems support for memristor-based accelerators," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, 2017, pp. 1–8.
[62] L. Gao, F. Alibart, and D. B. Strukov, "Analog-input analog-weight dot-product operation with Ag/a-Si/Pt memristive devices," in *Proc. IEEE/IFIP 20th Int. Conf. VLSI Syst. Chip (VLSI-SoC)*, 2012, pp. 88–93.
[63] L. Han, Z. Shen, Z. Shao, H. H. Huang, and T. Li, "A novel ReRAM-based processing-in-memory architecture for graph computing," in *Proc. IEEE 6th Nonvolatile Memory Syst. Appl. Symp. (NVMSA)*, 2017, pp. 1–6.
[64] J. Mustafa and R. Waser, "A novel reference scheme for reading passive resistive crossbar memories," *IEEE Trans. Nanotechnol.*, vol. 5, no. 6, pp. 687–691, Nov. 2006.
[65] S. Shin, K. Kim, and S.-M. Kang, "Analysis of passive memristive devices array: Data-dependent statistical model and self-adaptable sense resistance for RRAMS," *Proc. IEEE*, vol. 100, no. 6, pp. 2021–2032, Jun. 2012.
[66] Y. Deng *et al.*, "RRAM crossbar array with cell selection device: A device and circuit interaction study," *IEEE Trans. Electron Devices*, vol. 60, no. 2, pp. 719–726, Feb. 2013.
[67] Z. Tang, Y. Wang, Y. Chi, and L. Fang, "Comprehensive sensing current analysis and its guideline for the worst-case scenario of RRAM read operation," *Electronics*, vol. 7, no. 10, p. 224, 2018.

Wen Wen (S'17) received the B.E. and M.E. degrees from Southeast University, Nanjing, China, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA.

His current research interests include computer architecture and emerging nonvolatile memory technologies.

Lei Zhao received the B.S. and M.S. degrees in computer science from Northwestern Polytechnical University, Xi'an, China, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Computer Science Department, University of Pittsburgh, Pittsburgh, PA, USA, under the guidance of Prof. Y. Zhang.

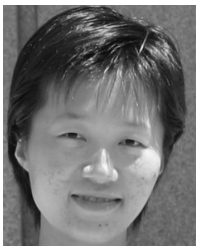
His current research interests include emerging memory technologies and acceleration techniques for machine learning applications with a focus on model security.



Youtao Zhang (M'04) received the Ph.D. degree in computer science from the University of Arizona, Tucson, AZ, USA, in 2002.

He is an Associate Professor with the Computer Science Department, University of Pittsburgh, Pittsburgh, PA, USA. His current research interests are in the areas of architectural support for security, new memory technologies, acceleration for machine learning, program analysis and optimization.

Dr. Zhang was a recipient of the U.S. NSF Career Award in 2005, the Distinguished Paper Award of the IEEE/ACM International Conference on Software Engineering Conference in 2003, and the Most Original Paper Award of the International Conference on Parallel Processing Conference in 2003. He is a member of ACM.



Jun Yang (SM'19) received the B.S. degree in computer science from Nanjing University, Nanjing, China, in 1995, and the Ph.D. degree in computer science from the University of Arizona, Tucson, AZ, USA, in 2002.

She is a Professor with the Electrical and Computer Engineering Department, University Pittsburgh, Pittsburgh, PA, USA. Her current research interests include GPU architecture, secure processor architecture, emerging nonvolatile memory technologies, and performance and reliability of memories.

Dr. Yang was a recipient of the U.S. NSF Career Award in 2008 and the Best Paper Awards from ICCD 2007 and ISLPED 2013.