Speeding Up Crossbar Resistive Memory by Exploiting In-memory Data Patterns

Wen Wen[†], Lei Zhao[§], Youtao Zhang[§], Jun Yang[†]
[†]Department of Electrical and Computer Engineering, [§]Department of Computer Science
University of Pittsburgh
[†]wew55@pitt.edu, [§]lez21@pitt.edu, [§]zhangyt@cs.pitt.edu, [†]juy9@pitt.edu

***Abstract*—Resistive Memory (ReRAM) has emerged as a promising non-volatile 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. ReRAM, when adopting crossbar architecture, has the smallest $4F^2$ planar cell size, which is ideal for constructing dense memory with large capacity. However, crossbar cell structure suffers from large sneak leakage and IR drop on long wires. To ensure operation reliability, ReRAM writes, in particular, RESET 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 paper, we study the correlation between the RESET latency and the number of cells in low resistant state (LRS) along bitlines, and propose to dynamically speed up ReRAM RESET operations for the rows that have small numbers of LRS cells. 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 RESET latency reduction, we employ data compression and row address dependent data layout to reduce LRS cells on bitlines. 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.

***Index Terms*—Resistive Memory, Data Pattern, Crossbar Array**

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 [13]. Recent studies have proposed to construct future large capacity main memory using emerging non-volatile memory (NVM) technologies, e.g., PCM (Phase Change Memory), STT-MRAM (Spin Transfer Torque Magnetic RAM), and ReRAM (Resistive Memory) [15], [20]–[24]. 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 and better density and scalability than STT-MRAM. When adopting crossbar architecture, ReRAM can achieve the smallest $4F^2$ planar cell size [21].

However, ReRAM crossbars suffer from large sneaky currents. 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 [18], 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 paper, we focus on mitigating the performance degradation from IR drop. We summarize our contributions as follows.

- We study the correlation between the RESET latency of a ReRAM row and the number of the cells in low resistance state (LRS) on selected bitlines. We propose 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.
  To the best of our knowledge, this paper is the first architectural innovation that exploits the bitline data patterns, e.g., the percentage of LRS cells, to speed up RESET operations in ReRAM crossbars.
- 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.
- 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 15.7% and 7.6%, compared to the baseline and the state-of-the-art crossbar design.

In the rest of the paper, we introduce the ReRAM basics and motivations in Section 2. We elaborate the design details in Section 3. We present the experiment methodology in Section 4 and discuss the experimental results in Section 5. We discuss additional related work in Section 6 and conclude the paper in Section 7.

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 (low resistant state) cells.

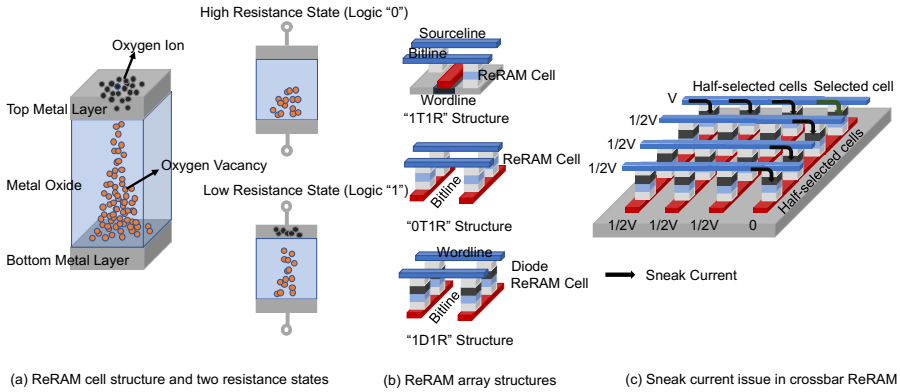


Fig. 1: The ReRAM basics: (a) the cell structure; (b) the three typical ReRAM array structures; and (c) the sneak current issue in ReRAM crossbar array.

A. ReRAM Basics

ReRAM is a promising non-volatile memory technology that stores data using cell resistance. As shown in Figure 1(a), a 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: a low resistance state (LRS) to represent logic '1' and a high resistance state (HRS) to represent logical '0'. To program a 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

Figure 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 paper, 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 64B and its 512 bits are saved in 64 mats (subarrays) with each subarray containing 8 bits, the same as that in [21]. These mats spread across 8 chips in one rank. To perform a RESET operation in a 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 bitlines 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 1(c).

- **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.

A cacheline write operation consists of two phases: a RESET phase to write all 0s and a SET phase to write all 1s. We adopt *DSGB* to improve write performance [21] and *flip-n-write* to only write modified cells [7]. Since SET operation takes shorter time than RESET operation [21], [22], [24], we focus on long latency RESET operations in the paper. The proposed scheme is applicable to the ReRAM structures that have comparable SET and RESET latencies.

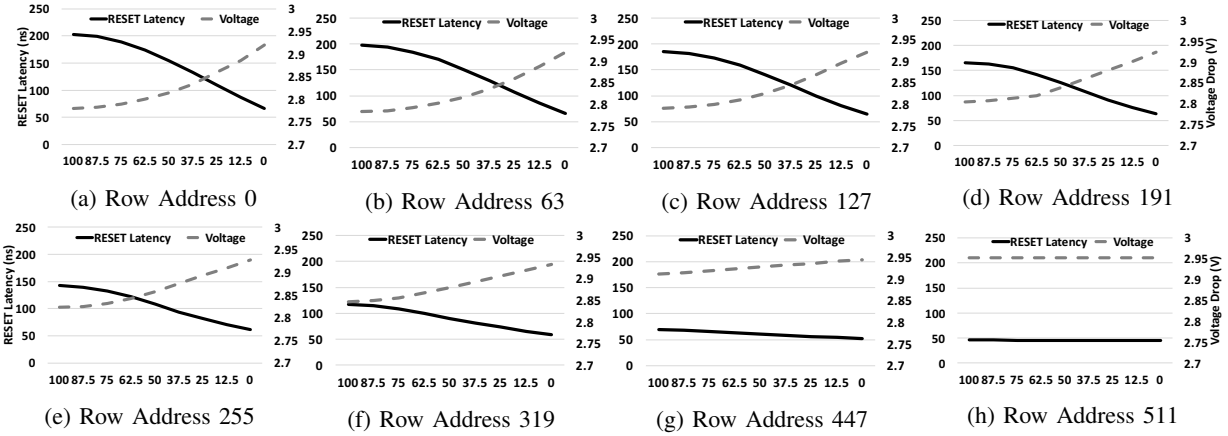
1) IR Drop Issue: Studies have shown that ReRAM crossbar, even adopting diode selectors, has the currents flowing through all cells — while the sneaky currents flowing through not-selected cells are negligible, those flowing through half-selected cells are not. The sneak currents introduce large voltage 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 [22].

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) The 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 Equation 1, as shown in recent studies [9], [21].

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

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 $10\times$ RESET latency increase [9].



(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

Fig. 2: 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.

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 [18].

Given one selected wordline and one selected bitline, we study the correlation among IR drop, the number of LRS cells, and RESET latency. Figure 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. We focus on bitline LRS cells and assume the worse case for the wordline in this paper. The impact from wordline tends to be smaller due to the adoption of DSGB [21] and each subarray saving 8 bits from one cacheline. We study the RESET latency in this paper, a similar observation for READ was reported in [5]. In the experiments, we adopted the Verilog-A model from [12] 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 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.

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

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.

A. An Overview

Figure 3 presents an overview of our proposed scheme. We assume that each cacheline has 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 [21]. 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. 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..512] into 8 subranges such that each 3-bit flag denotes a subrange, e.g., '000' denotes subrange [0..63] and '010' denotes subrange [128..191].

We exploit a runtime profiler that periodically detects the worst case bitline in each mat and then determines the wrost 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.

RESET latency optimization. To RESET a memory line, we fetch its `W-Flag` and `W-Cnt` to determine the appropriate tWR time for the RESET operation. For example, if `row 0`'s flag is '010', we use tWR=155ns instead of 203ns in the baseline.

We next elaborate the design details.

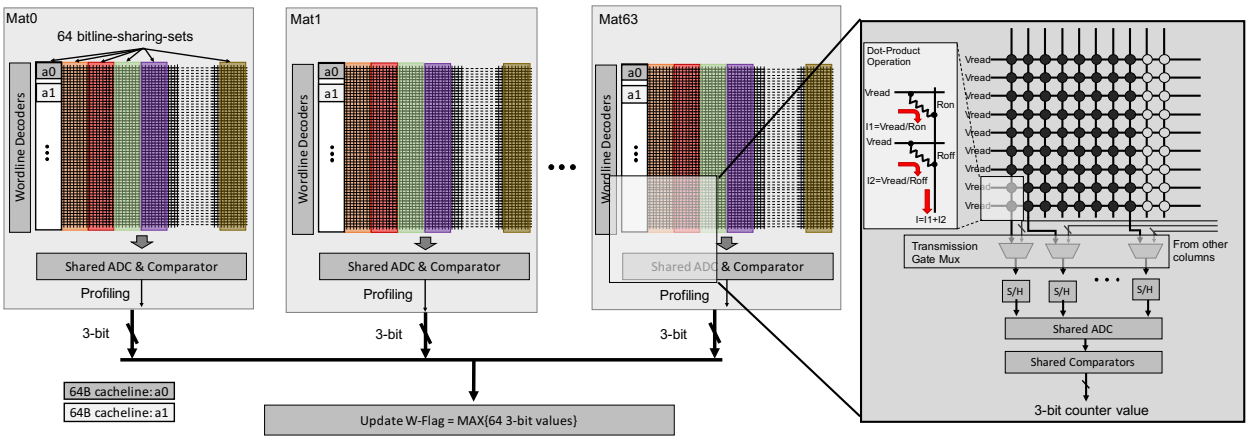


Fig. 3: An overview of the proposed architecture.

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 paper, we leverage the current aggregation feature of ReRAM crossbar array [11], which has been widely exploited for accelerating in-memory computation [4], [6], [17], [19].

Figure 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 row address. The row address determines the *bitline-sharing-set* in 64 mats. For each mat, all wordlines and the eight bitlines that belong to the *bitline-sharing-set* are activated for operation. This is similar to dot-product operation in [6].

As shown in the figure, all wordlines are applied with $\mathtt{V_{read}}$; the selected eight bitlines are applied with 0V; and all other bitlines are applied with $\mathtt{V_{read}}$ 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.

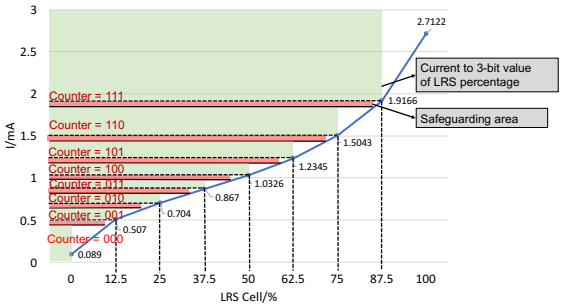


Fig. 4: The profiling current vs. LRS cell percentage in 512×512 ReRAM crossbar array.

We divide the range [0..512] into eight subranges with equal size (except the last one which has one more value). As shown in Figure 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 $0.1mA$ guard band 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, we tag a bitline 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, i.e., once `W-Cnt` overflows. An alternative design is to increment `W-Flag` if `W-Cnt` overflows. Clearly `W-Flag` would saturate after at most 512 writes to the set. However, normal writes not always introduce more LRS cells to the worst-case bitline, it is beneficial to periodically profile the set.

C. 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 [1] and fill in unused cells with 0s, i.e., reset them to HRS. 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×K (0≤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 0s, as shown in Figure 5.

D. Determine the RESET timing

At runtime, we use the physical address and the two flags `W-Flag` and `W-Cnt` to determine the appropriate tWR timing for the RESET operation.

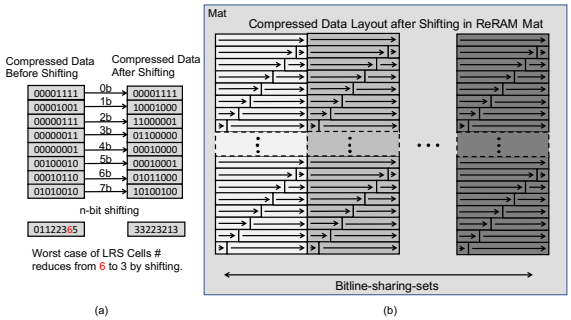


Fig. 5: Reducing LRS cells through data compress: (a) logic view; (b) shift in each mat.

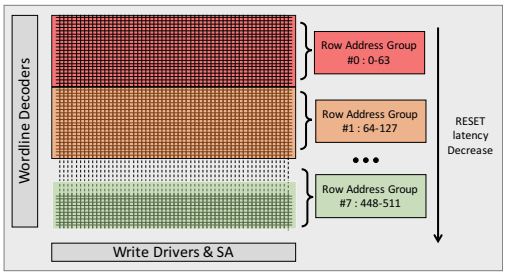


Fig. 6: The rows with different addresses are mapped to 8 groups with different worst-case RESET latencies.

The reason that we also use the row address is that, similar as that in [22], 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 Figure 6.

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.

TABLE II: The 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

E. 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 III summarizes the overheads for each ReRAM memory bank. We evaluated the power consumption and area by HSPICE simulation and NVSim [8] 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.

We followed recent studies [14], [17] 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 1.28GS/s sampling speed and introduces 50ns 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.

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 64B memory lines, or 32KB data. For a 8GB memory system, we need about 288KB storage to hold all flags. In this paper, 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.

TABLE III: Comparing the Profiling Overhead in One Bank

Comp.	Params	Spec	Power/Energy	Area (mm2)
ADC [14]	sampling speed resolution number	1.28GS/s 8-bit 8	24.48mW	0.012
S+H [17]	number	8×64	5uW	0.00002
ReRAM array	Mat number Mat size	1024 512×512	Profiling: 267.178pJ Read: 72.842pJ Leakage: 255.233mW	2.078

IV. Experimental Methodologies

To evaluate the effectiveness of our proposed design, in addition to the HSPICE modeling and simulation as introduced in Section 3.5, 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 [8] simulations into our architectural simulator to obtain the performance and memory energy efficiency results. We used Pintool to generate memory access traces from SPEC2006 [10], PARSEC [3] and BioBench [2] benchmark suites.

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.

In the paper, we 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 [22] 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.
- ALL — This scheme is built on top of CMP and includes all enhancements in the paper. In particular, it adopts a two dimensional tWR timing table (as shown in Table 2) in determining RESET latency.

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

V. Experiment Results

A. Memory Access Latency

Figure 7 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 ALL, respectively. Compared to RA, the proposed scheme ALL shows 53.5% more reduction. In summary, it is effective to reduce RESET latency by exploiting the number of LRS cells along bitlines.

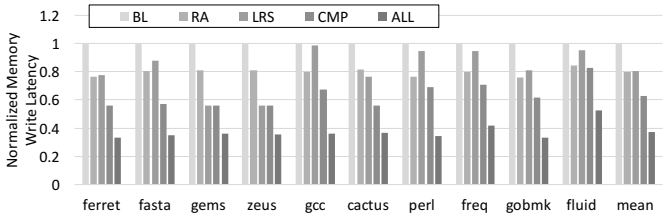


Fig. 7: The comparison of memory write latency.

The reduction of RESET latency leads to the reduction of memory read latency. Figure 8 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 ALL respectively. Our proposed ALL scheme shows a 27.6% more reduction over RA.

B. System Performance

We compared the performance when adopting different schemes and summarized the CPI (cycles-per-instruction) results in Figure 9. The results are normalized to BL. From the figure, the proposed scheme ALL achieves large performance improvements on memory intensive benchmarks, e.g., ferret and fasta_dna. On average, the scheme ALL achieves 20.5% and 14.2% performance improvements over BL and RA, respectively.

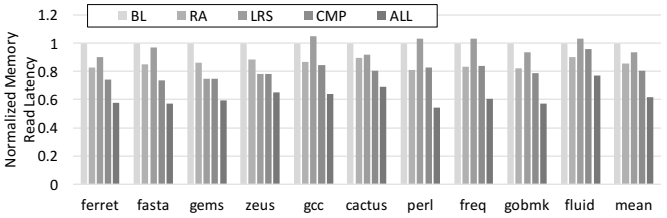


Fig. 8: The comparison of memory read latency.

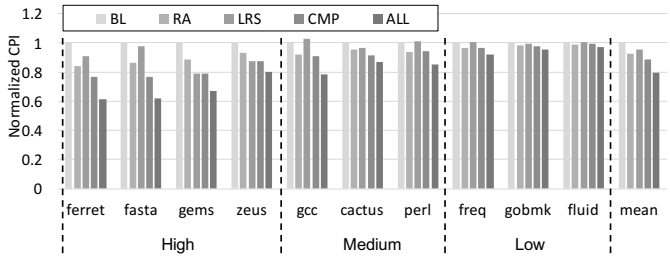


Fig. 9: The performance comparison. The benchmarks are categorized into High, Medium and Low memory intensity types based on RPKI and WPKI.

C. 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 Figure 10. The dynamic energy consumption has three major sources: read, write (including SET and RESET) energy and profiling overheads from our proposed schemes. While the ALL 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.

In summary, the scheme ALL achieves 15.7% and 7.6% dynamic energy reduction over BL and RA, respectively. The EDP results show that our proposed design can effectively improve the energy efficiency — ALL achieves 31.9% and 19.5% EDP improvements over BL and RA, respectively.

D. Sensitivity Study

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 ALL in Figure 11 (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.

Sensitivity to Mat Sizes. Figure 11 (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 ALL 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 ALL is slightly worse (only 1.6%) than RA for 256×256 mat size. This is because the profiling latency and power consumption are independent

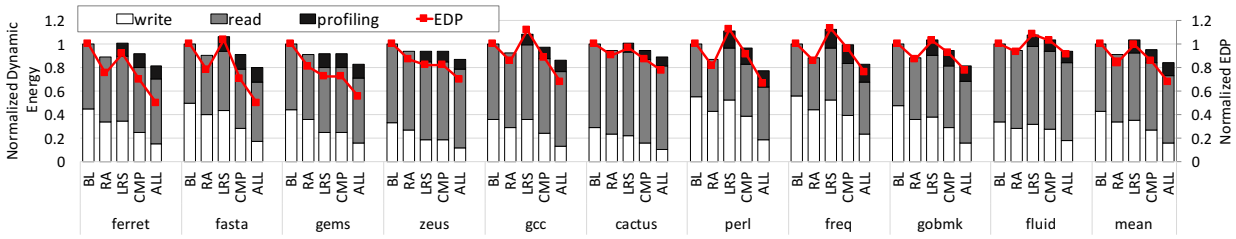


Fig. 10: The comparison of dynamic energy and Energy-Delay Product (EDP).

of mat size, which has a larger impact on smaller mats. 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.

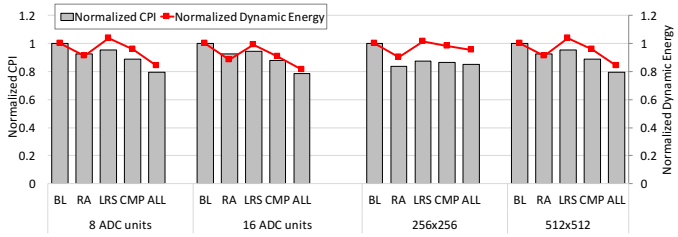


Fig. 11: The sensitivity of performance and memory dynamic energy consumption when using (a) different numbers of ADC units; and (b) different ReRAM mat sizes.

VI. Related Work

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 [21], [22], [24]. Xu *et al.* [21] 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.* [22] proposed to divide a crossbar array into several logical regions with different access latency, in order to exploit the discrepancy of RESET latency.

Current accumulation feature of ReRAM crossbar. Recent studies exploited the natural current accumulation feature of ReRAM crossbar architecture to implement dot-product analogy calculations [4], [6], [17], [19]. In this paper, we leverage this feature to profile and track the number of LRS cells along each bitline.

Data pattern in ReRAM crossbar. Chang *et al.* [5] presented a similar observation for read operation. Xu *et al.* [21] 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 *et al.* [16] analyzed the voltage drop and data patterns in ReRAM crossbar arrays without selectors.

VII. Conclusions

In this paper, based on the observation that the RESET latency strongly correlates to the number of cells in low resistant states (LRS) 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. 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.

Acknowledgment

This research is supported in part by NSF CCF-1535755 and NSF CCF-1617071.

References

[1] A. R. Alameldeen, *et al.* Frequent pattern compression: A significance-based compression scheme for l2 caches. In *TR Univ. of Wisconsin, Madison, TR#1500*. 2004.
[2] K. Albayraktaroglu, *et al.* Biobench: A benchmark suite of bioinformatics applications. In *ISPASS*, 2005.
[3] C. Bienia and K. Li. Parsec 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, 2009.
[4] M. N. Bojnordi and E. Ipek. Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *HPCA*, 2016.
[5] M.-F. Chang, *et al.* 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*, 2015.
[6] P. Chi, *et al.* Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *ISCA*, 2016.
[7] S. Cho and H. Lee. Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance. In *MICRO*, 2009.
[8] X. Dong, *et al.* Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012.
[9] B. Govoreanu, *et al.* 10× 10nm 2 hf/hfo x crossbar resistive ram with excellent performance, reliability and low-energy operation. In *IEDM*, 2011.
[10] J. L. Henning. Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 2006.
[11] M. Hu, *et al.* Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication. In *DAC*, 2016.
[12] Z. Jiang, *et al.* Verilog-a compact model for oxide-based resistive random access memory (rram). In *SISPAD*, 2014.
[13] U. Kang, *et al.* Co-architecting controllers and dram to enhance dram process scaling. In *The memory forum*, 2014.
[14] 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. *JSSC*, 2013.
[15] H. H. Li, *et al.* Looking ahead for resistive memory technology: A broad perspective on reram technology for future storage and computing. *IEEE Consumer Electronics Magazine*, 2017.
[16] J. Liang and H.-S. P. Wong. Cross-point memory array without cell selectors — device characteristics and data storage pattern dependencies. *IEEE Transactions on Electron Devices*, 2010.
[17] A. Shafiee,, *et al.* Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *ISCA*, 2016.
[18] M. Shevgoor, *et al.* Improving memristor memory with sneak current sharing. In *ICCD*, 2015.
[19] L. Song, *et al.* Pipelayer: A pipelined reram-based accelerator for deep learning. In *HPCA*, 2017.
[20] H.-S. P. Wong, *et al.* Metal–oxide rram. *Proceedings of the IEEE*, 2012.
[21] C. Xu, *et al.* Overcoming the challenges of crossbar resistive memory architectures. In *HPCA*, 2015.
[22] H. Zhang, *et al.* Leader: Accelerating reram-based main memory by leveraging access latency discrepancy in crossbar arrays. In *DATE*, 2016.
[23] L. Zhang, *et al.* Mellow writes: Extending lifetime in resistive memories through selective slow write backs. In *ISCA*, 2016.
[24] L. Zhao, *et al.* Constructing fast and energy efficient 1tnr based reram crossbar memory. In *ISQED*, 2017.