# Tiered-ReRAM: A Low Latency and Energy Efficient TLC Crossbar ReRAM Architecture

Yang Zhang, Dan Feng*, Wei Tong, Jingning Liu, Chengning Wang, Jie Xu
Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System
School of Computer Science and Technology, Huazhong University of Science and Technology,
Ministry of Education of China, *Co-corresponding author: Dan Feng (dfeng@hust.edu.cn)
Email:(youngzhang, dfeng, tongwei, jnliu, chengningwang, xujie_dsal)@hust.edu.cn

***Abstract*—Resistive Memory (ReRAM) is promising to be used as high density storage-class memory by employing Triple-Level Cell (TLC) and crossbar structures. However, TLC crossbar ReRAM suffers from high write latency and energy due to the IR drop issue and the iterative program-and-verify procedure. In this paper, we propose Tiered-ReRAM architecture to overcome the challenges of TLC crossbar ReRAM. The proposed Tiered-ReRAM consists of three components, namely Tiered-crossbar design, Compression-based Incomplete Data Mapping (CIDM), and Compression-based Flip Scheme (CFS). Specifically, based on the observation that the magnitude of IR drops is primarily determined by the long length of bitlines in Double-Sided Ground Biasing (DSGB) crossbar arrays, Tiered-crossbar design splits each long bitline into the near and far segments by an isolation transistor, allowing the near segment to be accessed with decreased latency and energy. Moreover, in the near segments, CIDM dynamically selects the most appropriate IDM for each cache line according to the saved space by compression, which further reduces the write latency and energy with insignificant space overhead. In addition, in the far segments, CFS dynamically selects the most appropriate flip scheme for each cache line, which ensures more high resistance cells written into crossbar arrays and effectively reduces the leakage energy. For each compressed cache line, the selected IDM or flip scheme is applied on the condition that the total encoded data size will never exceed the original cache line size. The experimental results show that, on average, Tiered-ReRAM can improve the system performance by 30.5%, reduce the write latency by 35.2%, decrease the read latency by 26.1%, and reduce the energy consumption by 35.6%, compared to an aggressive baseline.**

***Index Terms*—TLC crossbar ReRAM, IR drop, compression, Incomplete Data Mapping, flip scheme, write latency and energy**

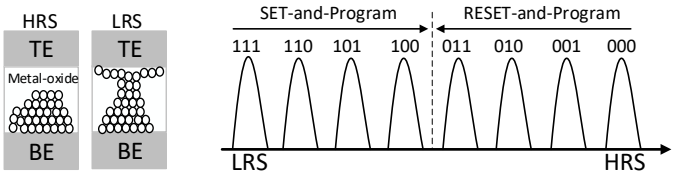
## I. Introduction

Modern data-intensive applications have exhibited increasing demand for large capacity memory, such as graphical games and big data analytics. However, DRAM, the *de facto* choice for constructing main memory, faces short refreshing interval, low density and scalability challenges. As indicated in ITRS, the scaling path of DRAM beyond 16nm is not clear [1]. Recently, Non-Volatile Memories (NVMs), e.g., Phase Change Memory(PCM), Spin-Transfer Torque Magnetic RAM (STT-MRAM) and Resistive Memory (ReRAM), have emerged as potential candidates for the storage-class memory due to their good scalability, high density, low standby power and non-volatility [2], [3], [4], [5], [6]. Among these candidates, ReRAM has become more promising due to its higher density and lower power consumption [4], [5], [6].

ReRAM can store three bits into a single cell with Triple-Level Cell (TLC) structure to improve the density [7]. Moreover, by employing the unique crossbar array structure, ReRAM can be constructed with the smallest $4F^2$ planar cell size [8], and thus higher density can be achieved. However, TLC crossbar ReRAM also faces many challenges in terms of performance and energy consumption. First, the crossbar structure suffers from an IR drop issue due to sneak currents and wire resistance, which causes high leakage energy and non-uniform access latency in crossbar arrays. Unfortunately, conventional ReRAM writes conservatively use the worst-case access latency of all cells, resulting in significant performance degradation. Second, the iterative program-and-verify (P&V) procedure of TLC ReRAM incurs high write latency and energy. Recent study has demonstrated that the write latency of a 4Mb Single-Level Cell (SLC) ReRAM is only $7.2ns$, while the write latency of Multi-Level Cell (MLC) ReRAM is $160ns$ and TLC ReRAM has much higher write latency [9]. The write energy of TLC ReRAM is also seven times higher than that of SLC ReRAM [10], [11]. Therefore, the high write latency and energy have become the greatest design concerns in TLC crossbar ReRAM-based memory systems.

Recently, many techniques have been proposed to optimize the TLC crossbar ReRAM. Double-Sided Ground Biasing (DSGB) design [4] has been proposed to reduce the IR drops along wordlines by applying another ground on the other side of the selected wordline. However, the magnitude of IR drops depends of both wordlines and bitlines [6]. Due to the long length and large wire resistance of bitlines, the IR drops along bitlines are still large, leading to significant performance degradation and energy waste. Incomplete Data Mapping (IDM) [10] has been proposed to reduce the write latency and energy of TLC ReRAM by eliminating certain high-latency and high-energy states. 0-Dominated Flip Scheme (0-DFS) [12] has also been proposed to increase the number of high resistance cells in crossbar arrays and reduce the leakage energy by flipping the written data with the additional flip flag bits. However, both IDM and 0-DFS significantly sacrifice the capacity of TLC ReRAM and the two techniques are limited by the space overhead.

In this paper, we propose **Tiered-ReRAM**, a low latency




(a) HRS/LRS (b) TLC resistance distribution

Fig. 1: ReRAM cell structure.

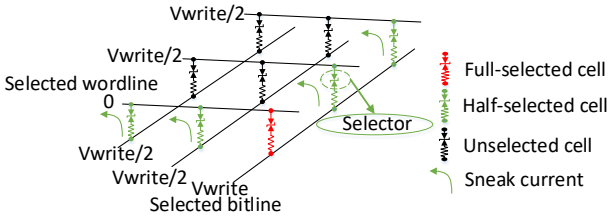



Fig. 2: RESET operation in 1S1R crossbar structure.

and energy efficient TLC crossbar ReRAM architecture, to overcome the challenges of TLC crossbar ReRAM. Tiered-ReRAM architecture consists of three components, including the Tiered-crossbar design, Compression-based Incomplete Data Mapping (CIDM), and Compression-based Flip Scheme (CFS): (1) Based on the observation that the magnitude of IR drops is primarily determined by the long length of bitlines in DSGB crossbar arrays, Tiered-crossbar design splits each long bitline into the near and far segments by an isolation transistor, allowing the near segment to be accessed with decreased IR drops and latency/energy. Similar to many prior works [13], [14], [15], [16], [17], Tiered-crossbar design also remaps hot data to the near segments and cold data to the far segments, which significantly improves the access performance. (2) We also observe that most cache lines can be compressed and the saved space by compression varies greatly. On the other hand, different IDMs have different tradeoffs in space overhead and write latency/energy. The IDM that eliminates more states to encode sacrifices more capacity, but achieves lower write latency and energy. Therefore, in performance-sensitive near segments, CIDM delicately applies the compression technique in conjunction with IDM. CIDM dynamically selects the most appropriate IDM for each cache line according to the saved space by compression, which further reduces the write latency and energy with insignificant space overhead. (3) Although programming high resistance TLC cells causes higher write latency (Indicated in Table I), the high resistance cells in crossbar arrays can effectively reduce the sneak currents and leakage energy according to Ohm's law. Flip schemes (e.g., 0-DFS [12] and CAFO [18]) can be used to increase the number of high resistance cells in crossbar arrays. We observe that different word-size flip schemes have different tradeoffs in effects and space overhead. The flip scheme that uses smaller word size achieves more high resistance cells and lower leakage energy, but results in higher space overhead. Therefore, in performance-insensitive far segments, CFS subtly combines the compression technique with the flip scheme. CFS dynamically selects the most appropriate flip scheme for each cache line according to the saved space by compression, which ensures more high resistance cells written in crossbar arrays and effectively reduces the leakage energy with insignificant space overhead. For each compressed cache line, the selected IDM or flip scheme is applied on the condition that the total encoded data size will never exceed the original cache line size. The contributions of this paper include:

- Based on the observation that the magnitude of IR drops is primarily determined by the long length of bitlines in DSGB crossbar arrays, we propose a microarchitectural design called *Tiered-crossbar* to split each long bitline into the near and far segments by an isolation transistor, which effectively mitigates the IR drop issue and enables low latency/energy ReRAM.
- We implement CIDM and CFS techniques in the near and far segments, respectively, by dynamically selecting the most appropriate IDM or flip scheme for each cache line according to the saved space by compression, which further reduces the write latency and energy of TLC crossbar ReRAM.
- We evaluate the proposed Tiered-ReRAM with comparison to an aggressive baseline. The experimental results show that, on average, Tiered-ReRAM can improve the system performance by 30.5%, reduce the write latency by 35.2%, decrease the read latency by 26.1%, and reduce the energy consumption by 35.6%.

## II. Background

### A. ReRAM Cell Structure

A ReRAM cell consists of a metal-oxide layer sandwiched between a top electrode (TE) and a bottom electrode (BE), as shown in Figure 1a. The state of a ReRAM cell is represented by its resistance value. For a SLC ReRAM cell, the high resistance state (HRS) and low resistance state (LRS) are used to represent logic 0 and logic 1, respectively. In order to switch the resistance state of a ReRAM cell, an external voltage with specified polarity, magnitude and duration should be applied to the cell. The switching from HRS to LRS is referred to as a SET operation and the switching from LRS to HRS is referred to as a RESET operation. Due to large resistance differences between HRS and LRS (Resistance ratio of HRS to LRS can exceed 1000), ReRAM has the TLC feature to divide the wide range resistance into eight levels for storing three bits into a single cell [7], [10], as shown in Figure 1b. Compared with SLC ReRAM, TLC ReRAM offers higher data density.

### B. ReRAM Array Structure

ReRAM array structure can be classified into three types: 1T1R, crossbar and 1TnR. In 1T1R structure, each cell has a dedicated access transistor so that it can be accessed independently without disturbance. However, since the size of an access transistor is typically much larger than that of a ReRAM cell, 1T1R structure significantly reduces the area efficiency.

In crossbar structure, all cells are interconnected to each other without access transistors and a cell only occupies

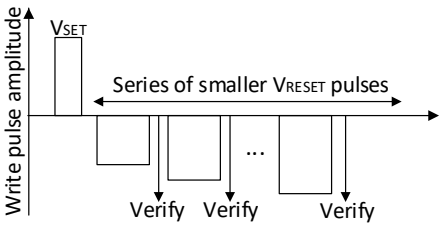


Fig. 3: SAP scheme for TLC ReRAM.

an area of $4F^2$, which is the smallest planar cell size [8]. Based on whether integrating a dedicated selector into each cell, the crossbar structure can be classified into 1S1R and 0T1R structures. Figure 2 shows the 1S1R crossbar structure. Compared with 0T1R cells, 1S1R cells can achieve the same cell size with higher nonlinearity and smaller sneak currents, which enables the fabrication of large crossbar arrays.

In 1TnR structure, n cells share 1 access transistor. 1TnR structure is a tradeoff between 1T1R and 0T1R structures. Considering the lower fabrication cost and better scalability, 1S1R crossbar structure is more suitable for constructing the high density storage-class memory. In this work, we select the $512 \times 512$ 1S1R crossbar ReRAM as our baseline, which has been widely used in community [4], [5].

### C. IR Drop Issue of Crossbar

To RESET or SET a cell in the crossbar array, the bitline and wordline connected to the target cell should be activated with the proper potential ($\pm V_{write}$). In addition, the unselected bitlines and wordlines are set to $V_{write}/2$ to avoid disturbing other cells in the array. Figure 2 shows the RESET operation in the crossbar array. The target cell is applied with full voltage ($V_{write}$), referred to as *full-selected cell*. Other cells on the selected bitline and wordline are half biased at $V_{write}/2$, referred to as *half-selected cells*. The cells on the unselected bitlines and wordlines are referred to as *unselected cells*. The currents flowing across half-selected cells are commonly referred to as *sneak currents*. The sneak currents and wire resistance introduce large voltage reduction along bitlines and wordlines, referred to as *IR drop issue*.

The IR drop issue reduces the voltage drop across the target cell. Unfortunately, the RESET switching time of a ReRAM cell is exponentially inverse to the voltage drop across the cell [4], [19]. Therefore, the IR drop issue significantly increases the RESET latency. Since ReRAM cells at different locations of the crossbar array have different IR drops, this characteristic causes non-uniform access latency in crossbar arrays. Worse still, the IR drop issue leads to high leakage energy due to the sneak currents of LRS half-selected cells. Recent study has demonstrated that in a $100 \times 100$ crossbar array, only about 1% of the total energy is consumed by the access of the target cell and about 97% of the total energy is dissipated by the sneak currents of LRS half-selected cells [20].

### D. Iterative Program-and-verify of TLC ReRAM

Due to the process variation and statistical characteristics of TLC ReRAM, it is difficult to apply a generic scheme to precisely program a cell into a certain resistance range. Instead, program-and-verify (P&V) is commonly used for TLC ReRAM programming. The P&V programming can either start from a SET or RESET operation, followed by series of smaller $V_{RESET}$ or $I_{SET}$ pulses. Each $V_{RESET}$ or $I_{SET}$ is followed by a read operation to verify the state of the cell. If the resistance of the cell reaches the target range, the write operation terminates. Figure 3 illustrates the P&V programming procedure starting from a SET operation. The iterative P&V procedure results in high write latency and energy. SET-and-Program (SAP) and RESET-and-Program (RAP) schemes [10] have been proposed to reach the target states in fewer iterations, which benefits the write latency and energy. If the most significant bit (MSB) of the target state is '1', SAP scheme is used to reach the target states. Otherwise, RAP is applied, as shown in Figure 1b.

TABLE I: Iterations, Latency and Energy of TLC Writes

| Target states | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|---|
| Iterations | 1.21 | 5.27 | 10.1 | 15 | 14.3 | 9.83 | 4.68 | 1.52 |
| Latency (ns) | 14.2 | 95.4 | 192 | 290 | 383 | 338.3 | 286.8 | 255.2 |
| Energy (pJ) | 1.8 | 13.4 | 24.3 | 46.8 | 94 | 66.4 | 41.1 | 33.6 |

However, even with RAP and SAP schemes, the write latency and energy of TLC crossbar ReRAM are still high. The number of P&V iterations is highly dependent on the data written into the cell. Programming some states (e.g., states '100' and '011') requires more iterations, resulting in high write latency and energy. On the other hand, TLC writes with $V_{RESET}$ (e.g., '000') lead to higher latency/energy because the RESET operation is more sensitive to the IR drop issue. Table I shows the worst-case iterations, latency and energy for programming different TLC states in DSGB-based crossbar arrays. The parameters are obtained from Xu's works [4], [7].

## III. Motivation

### A. IR Drops in DSGB Design

To reduce the IR drops in crossbar arrays, Double-Sided Ground Biasing (DSGB) design has been widely adopted in community [4], [5]. DSGB applies another ground on the other side of the selected wordline. By doing so, the length of the worst-case IR drop path has been reduced and the IR drops along wordlines have been significantly mitigated. However, the magnitude of IR drops depends of both wordlines and bitlines [6]. Due to the long length and large wire resistance of bitlines, the IR drops along bitlines are still large, resulting in significant performance degradation and energy waste. Unfortunately, most prior studies [5], [13], [14] focus on leveraging the non-uniform access latency in crossbar arrays caused by IR drops, failing to optimize the IR drops from the source.

### B. Saved Space by Compression Varies

Compression techniques are commonly used to save the storage space [21], [22], [23], [24]. Frequent pattern compression (FPC) [25] and base-delta-immediate compression [26] are two typical compression techniques. FPC is evaluated in this work due to its high performance and low overhead. FPC can be used for 32-bit or 64-bit words. Each compressed word

TABLE II: The 64-Bit FPC Patterns with 3-Bit Prefix (Indicated in Red)

| Prefix | Pattern Encoded | Example | Compressed Example | Encoded Size | Saved Space |
|---|---|---|---|---|---|
| 000 | Zero run | 0x0000000000000000 | 0x0 | 3 bits | 61 bits |
| 001 | 8-bit sign extended | 0x000000000000007F | 0x17F | 11 bits | 53 bits |
| 010 | 16-bit sign extended | 0xFFFFFFFFFFFFB6B6 | 0x2B6B6 | 19 bits | 45 bits |
| 011 | Half-word sign extended | 0x0000000076543210 | 0x376543210 | 35 bits | 29 bits |
| 100 | Half-word, padded with a zero half-word | 0x7654321000000000 | 0x476543210 | 35 bits | 29 bits |
| 101 | Two half-words, each a byte sign extended | 0xFFFFBEEF00003CAB | 0x5BEEF3CAB | 35 bits | 29 bits |
| 110 | Word consisting of four repeated double bytes | 0xCAFECAFECAFECAFE | 0x6CAFE | 19 bits | 45 bits |

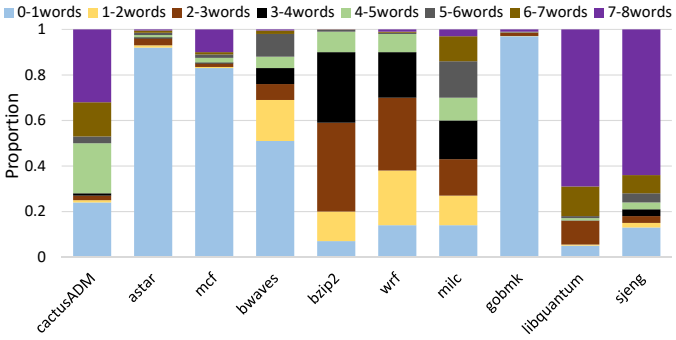



Fig. 4: The distribution of compressed cache line sizes.

is stored with a 3-bit prefix. By employing FPC, a 64-bit word can be compressed to 3, 11, 19, or 35 bits, and thus 61, 53, 45 or 29 bits storage space can be saved, as shown in Table II. For a cache line composed of eight 64-bit words, each word is compressed separately and the saved space varies. The saved space of the cache line may range from 0 to 488 bits. To quantitatively show the distribution of compressed cache line sizes, we implement FPC for the SPEC CPU2006 benchmarks in our architectural simulator (Detailed description in Section V). As shown in Figure 4, the compressed cache line sizes vary greatly. Some cache lines can be compressed to smaller than one word and more than seven words storage space can be saved. While some cache lines have more than seven words after compression, and the saved space is less than one word.

### C. Tradeoffs in Different IDMs

Incomplete Data Mapping (IDM) [10] can reduce the write latency and energy of TLC ReRAM by mapping only part of TLC ReRAM states into binary digits. IDM uses $q$ states out of a $p$-state cell to encode, where $q < p$. Then $r$ $q$-state cells are converted into k binary digits, where $q^r \approx 2^k$. This scheme is referred to as *IDM((p,q),r)*, as shown in Figure 5. Different from IDM, Complete Data Mapping (CDM) uses all the $p$ states to encode. Compared to CDM, IDM achieves lower write latency and energy by eliminating certain high-latency and high-energy states of TLC ReRAM, but sacrifices the capacity of TLC ReRAM.

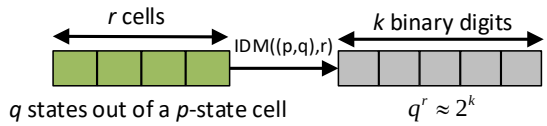



Fig. 5: *IDM((p,q),r)* scheme [10].

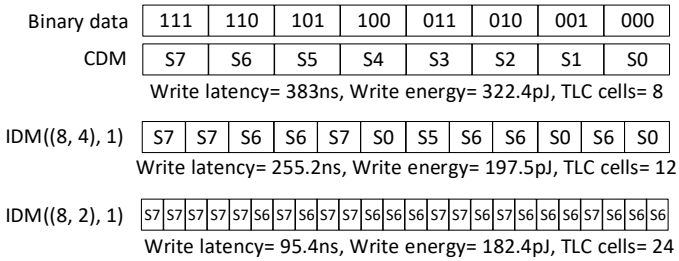



Fig. 6: An example for CDM, IDM((8,4),1) and IDM((8,2),1).

Different IDMs have different tradeoffs in space overhead and write latency/energy. The more states the IDM eliminates, the lower write latency/energy the TLC ReRAM achieves, and the larger capacity the TLC ReRAM sacrifices. Figure 6 is an example to show the tradeoffs in different IDMs. We assume the binary data to write are '111', '110', '101', '100', '011', '010', '001' and '000'. When using the CDM encoding, the eight resistance states ('S7', 'S6', 'S5', 'S4', 'S3', 'S2', 'S1' and 'S0') can store all the binary data and only 8 TLC cells are required. When applying the IDM((8,4),1) encoding, four latency/energy critical states ('S4', 'S3', 'S2' and 'S1') are eliminated and 12 TLC cells are required to store all the binary data. Although IDM((8,4),1) consumes 50% more capacity than CDM, it reduces the write latency and energy by 33.4% and 38.7%, respectively. When employing the IDM((8,2),1) encoding, only two low latency/energy states ('S7' and 'S6') are used and 24 TLC cells are needed to store all the binary data. Compared with CDM, IDM((8,2),1) costs twice more capacity, however it reduces the write latency and energy by 75.1% and 43.4%, respectively. Therefore, the IDM that eliminates more states to encode can sacrifice more capacity for more write latency/energy reduction.

### D. Tradeoffs in Different Word-size Flip Schemes

High resistance cells in crossbar arrays can effectively reduce the sneak currents and leakage energy according to Ohm's law. To ensure more high resistance TLC cells in crossbar arrays, we should guarantee that more Most Significant Bits (MSBs) of TLC cells are '0' during the write operation. Flip Schemes can be used to increase the number of '0' MSBs, such as 0-Dominated Flip Scheme (0-DFS) [12] and CAFO [18]. 0-DFS is a row-only flip scheme. 0-DFS flips the data with a flip flag bit '0' if the number of 0s in the data is smaller than or equal to N/2, where N is the word size of the 0-DFS. N can be 2, 4, 8 or 16. Different from 0-DFS, CAFO can simultaneously flip the rows and columns to achieve the most

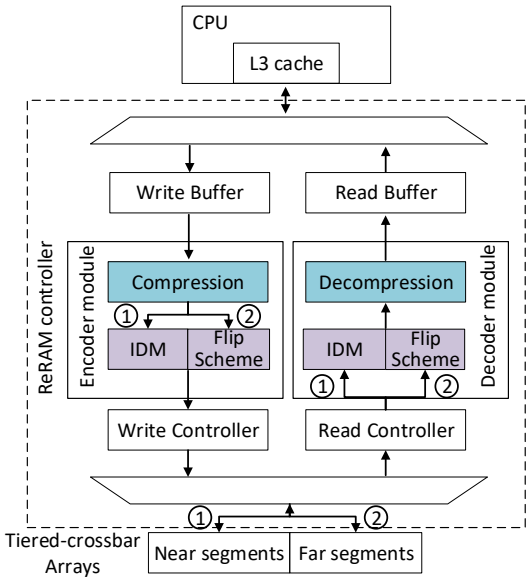



Fig. 7: Overview of Tiered-ReRAM architecture.

'0' MSBs. However, CAFO has much higher implementation overhead than 0-DFS. Thus, we evaluate 0-DFS in this work.

Different word-size 0-DFSs have different tradeoffs in effects and space overhead. The smaller word size the 0-DFS uses, the more '0' MSBs the 0-DFS achieves, and the larger space overhead the 0-DFS causes. If the word size is 2-bit, every 2 bits require a flip flag bit and the 0-DFS can achieve the most '0' MSBs. But the space overhead of the 2-bit word-size 0-DFS is 50%. If the word size is 16-bit, many data patterns that have more than eight 0s can't be flipped. In this case, the effect of the 0-DFS is the worst and the least '0' MSBs can be achieved. However, the space overhead of the 16-bit word-size 0-DFS is only 6.25%. Therefore, the 0-DFS that uses smaller word size can achieve more '0' MSBs with higher space overhead.

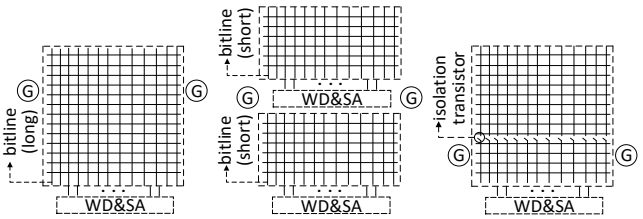
## IV. Tiered-ReRAM Architecture

### A. Overview

In this paper, we propose Tiered-ReRAM architecture to reduce the write latency and energy of TLC crossbar ReRAM. Figure 7 illustrates the overview of the proposed design. Tiered-ReRAM consists of three components, including the Tiered-crossbar design, Compression-based IDM (CIDM) and Compression-based Flip Scheme (CFS). The Tiered-crossbar design is performed in the ReRAM array level, aiming to fundamentally optimize the IR drops along bitlines based on DSGB design. CIDM is implemented in the ReRAM controller as Path ①, which delicately combines the compression technique with IDM and further reduces the write latency/energy with insignificant space overhead. CFS is performed in the ReRAM controller as Path ②, which applies the compression technique in conjunction with the flip scheme and reduces the leakage energy with insignificant space overhead. Next, we elaborate the design details of Tiered-ReRAM architecture.

TABLE III: Parameters in Our ReRAM Circuit Model

| Metric | Description | Value |
|---|---|---|
| A | Crossbar array size: A $wordlines\times$ A $bitlines$ | 512 |
| $R_{wire}$ | Wire resistance between adjacent cells | $2.82\Omega$ |
| $R_{LRS}$ | Resistance of a LRS cell | $100K\Omega$ |
| $K_r$ | Nonlinearity of a LRS ReRAM device | 10 |
| $K_s$ | Nonlinearity of the selector | 3000 |
| $V_W$ | Full selected voltage during write | 3.2V |
| $C_R$ | Capacity ratio of near segments to far segments | 1:3 |
| $-$ | Voltage biasing scheme | DSGB |




(a) DSGB (b) Latency Opt. (c) Tiered-crossbar

Fig. 8: Comparison among different crossbar designs.

### B. Tiered-crossbar Design

Double-Sided Ground Biasing (DSGB) design [4] uses another ground on the other side of the selected wordline to reduce IR drops along wordlines (Figure 8a). However, due to the long length and large wire resistance of bitlines, IR drops along bitlines are still large and the write latency/energy is still high. Shorter bitlines have the smaller wire resistance and IR drops, resulting in decreased write latency/energy, but require lots of write drivers (WD) and sense amplifiers (SA) for a given ReRAM capacity (Figure 8b).

Based on DSGB design, to mitigate IR drops along bitlines with insignificant cost, we propose a microarchitectural design called *Tiered-crossbar*. Tiered-crossbar splits each long bitline into two shorter segments using an *isolation transistor*, as shown in Figure 8c. The segment directly connected to the WD/SA is referred to as the *near segment*, and the other segment is referred to as the *far segment*. To access a ReRAM cell in the near segment, the isolation transistor on the selected bitline is turned off, so that only the bitline in the near segment incurs IR drops. Therefore, cells in the near segment have smaller IR drops and lower write latency/energy. On the other hand, to access a cell in the far segment, the isolation transistor on the selected bitline is turned on. In this case, the entire length of the bitline is connected to the WD/SA, similar to DSGB design (Figure 8a). Therefore, cells in the far segment have the same write latency/energy as DSGB design (Assume the resistance of the isolation transistor is negligible when it's turned on). Compared with the latency optimized crossbar array (Figure 8b), Tiered-crossbar array can decrease the additional transistors by 90.9% (WD and SA require 11 transistors per nanowire).

In Tiered-crossbar design, the capacity ratio of the near segments to the far segments is 1:3. To quantitatively show the effectiveness of Tiered-crossbar design, we build a detailed circuit model for the $512 \times 512$ TLC crossbar array according to Kirchhoff's Current Law [6], [12], [27]. The key parameters

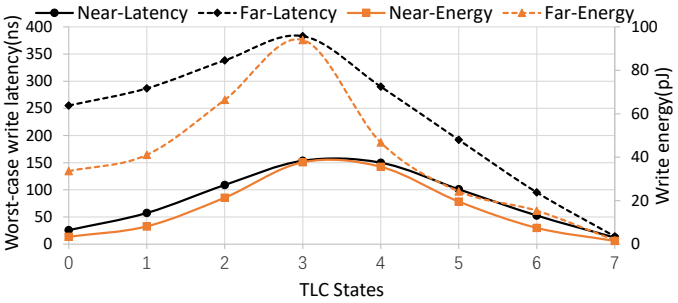



Fig. 9: The worst-case write latency/energy of the near and far segments in Tiered-crossbar design.

TABLE IV: The Most Appropriate IDM

| Saved space (bit) | Encoding method | 2-bit IDM flag |
|---|---|---|
| [341, 488] | IDM((8,2),1) | 11 |
| [170, 341) | IDM((8,4),1) | 10 |
| [85, 170) | IDM((8,6),2) | 01 |
| [0, 85) | CDM | 00 |

derived from the HfOx-based cells [19] and IBMs MIEC device [28] are presented in Table III. We obtain the worst-case voltage drops of the near and far segments from the circuit simulation. Then we get the worst-case write latency/energy according to the relationship between the voltage drop and write latency/energy [19]. Figure 9 illustrates the worst-case write latency/energy of the near and far segments in Tiered-crossbar design. The results show that compared to the far segments, the near segments can achieve 60% write latency reduction and 58% write energy reduction. Therefore, Tiered-crossbar design allows the near segments to be accessed with decreased latency and energy. In addition, similar to many prior works [13], [14], [15], [16], [17], [29], [30], [31], Tiered-crossbar design also remaps hot data to the near segments and cold data to the far segments, which significantly improves the access performance. In this work, we adopt the dynamic mapping method [13] to improve the access performance.

### C. Compression-based IDM

With the frequent pattern compression (FPC) technique [25], the saved space of a cache line may range from 0 to 488 bits. According to the saved space of each cache line, there exists the most appropriate IDM for the compressed data. For example, when IDM((8,2),1) is applied for the cache line that saves more than 340 bits by compression, the total encoded data size will not exceed the original cache line size. Moreover, IDM((8,2),1) can achieve more write latency/energy reduction than other IDMs. In this case, IDM((8,2),1) is the most appropriate encoding. However, when IDM((8,2),1) is applied for the cache line that only saves 170 to 340 bits by compression, the total encoded data size will exceed the original cache line size, resulting in high space overhead. Instead, IDM((8,4),1) is the most appropriate encoding in this case. To make full use of the saved space by compression for more latency/energy reduction, we should dynamically select the most appropriate IDM for each cache line according to the saved space by compression.

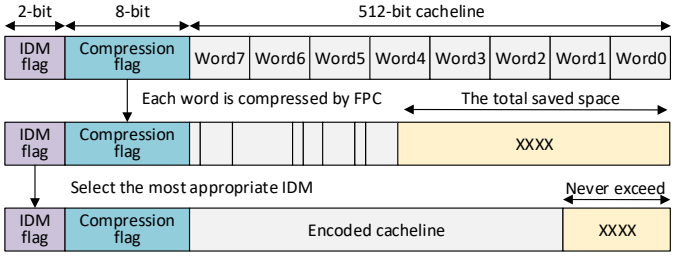



Fig. 10: Encoding procedure of CIDM.

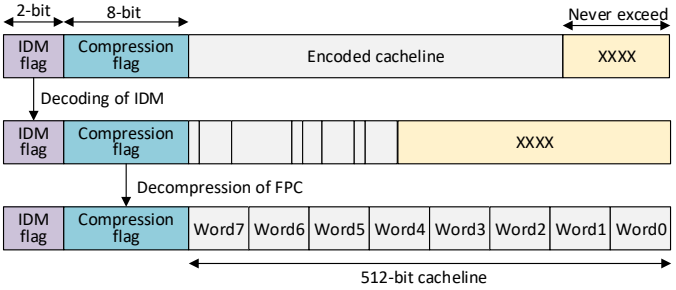



Fig. 11: Decoding procedure of CIDM.

Considering that hot data in the near segments of Tiered-crossbar arrays are sensitive to the access performance, we propose *compression-based IDM* (*CIDM*) in the near segments to further reduce the write latency/energy. CIDM dynamically selects the most appropriate IDM for each cache line according to the saved space by compression. Although a cache line is compressed word by word, the write latency of the cache line is determined by the slowest cell. Therefore, we implement CIDM at the cache line granularity. After the compression of the cache line, the saved space of each word is added up to calculate the total saved space. Then CIDM selects the IDM that eliminates as many latency/energy critical states as possible on the condition that the total encoded data size will never exceed the original cache line size. Thus, the compressed cache lines in the near segments are encoded through different IDMs. Table IV presents the most appropriate IDM for each cache line according to the saved space by compression. For simplicity, we evaluate three kinds of IDMs in this work, e.g., IDM((8,2),1), IDM((8,4),1) and IDM((8,6),2). A 2-bit IDM flag is required for each cache line to denote the encoding method. '11', '10', '01' and '00' are used to represent IDM((8, 2), 1), IDM((8, 4), 1), IDM((8, 6), 2) and CDM, respectively. In addition, each word needs one bit to indicate whether the word is compressed or not, e.g., '1' for the compressed word and '0' for the uncompressed word. Therefore, an additional 8-bit compression flag is required for each 8-word cache line.

CIDM consists of the CIDM encoder module on the write path and the CIDM decoder module on the read path. As shown in Figure 7, the CIDM encoder and decoder modules are embedded inside the ReRAM controller as Path ①.

***Write Path.*** When the near segments receive a write request from the processor, the CIDM encoder works as Figure 10

TABLE V: The Most Appropriate 0-DFS

| Saved space (bit) | Encoding method | 3-bit 0-DFS flag |
|---|---|---|
| [74, 488] | 2-bit word-size 0-DFS | 000 |
| [40, 74) | 4-bit word-size 0-DFS | 001 |
| [21, 40) | 8-bit word-size 0-DFS | 010 |
| [11, 21) | 16-bit word-size 0-DFS | 011 |
| [0, 11) | Without 0-DFS | 100 |

illustrates. First, the incoming cache line is passed through the compression logic to attempt data compression. Each word is compressed with FPC technique. The 8-bit compression flag is set to denote whether each word is compressed or not. The eight compressed words are stored contiguously and then the total saved space of the eight words is calculated. Second, the compressed cache line is encoded with the most appropriate IDM according to the total saved space by compression. The 2-bit IDM flag is set to represent the encoding method. The uncompressible cache lines are directly sent to the write circuit without the compression and IDM encoding.

***Read Path.*** When the near segments receive a read request from the processor, the CIDM decoder works as Figure 11 depicts. The 512-bit cache line with the corresponding IDM flag and compression flag is first read out. Then the 512-bit cache line is decoded by the IDM decoder module according to the IDM flag. After that, the decoded cache line is decompressed word by word according to the compression flag and the prefix of each word.

### *D. Compression-based Flip Scheme*

Although programming high resistance TLC cells causes higher write latency (Indicated in Table I), the high resistance TLC cells in crossbar arrays can effectively reduce the sneak currents and leakage energy. 0-Dominated Flip Scheme (0-DFS) [12] can be used to ensure more high resistance TLC cells in crossbar arrays by increasing the number of '0' MSBs during the write operation. However, different word-size 0-DFSs have different tradeoffs in effects and space overhead. The 0-DFS that uses smaller word size can achieve more '0' MSBs with higher space overhead. With the frequent pattern compression (FPC) technique [25], the saved space of a cache line may range from 0 to 488 bits. According to the saved space of each cache line, there exists the most appropriate 0-DFS for MSBs of the compressed data. For example, when the cache line saves more than 73 bits by compression, the size of the compressed cache line is smaller than 439 bits. For the compressed cache line, there are at most 146 MSBs to be written into TLC cells. In this case, the 2-bit word-size 0-DFS is the most appropriate encoding method because the 2-bit word-size 0-DFS can achieve the most '0' MSBs. Besides, the total encoded data size will not exceed the original cache line size. However, when the cache line only saves 11 to 20 bits by compression, the data size of the 2-bit word-size 0-DFS encoding will exceed the original cache line size, leading to high space overhead. Instead, the 16-bit word-size 0-DFS is the most appropriate encoding method in this case. To make full use of the saved space by compression for more leakage energy reduction, we should dynamically select the most appropriate

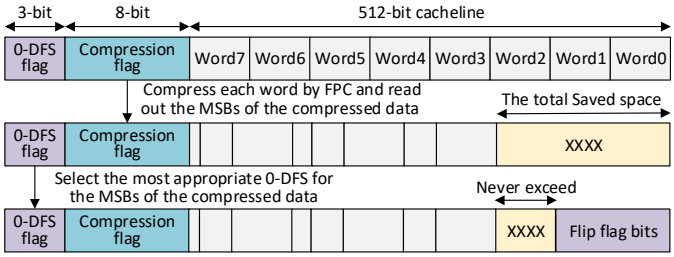



Fig. 12: Encoding procedure of CFS.

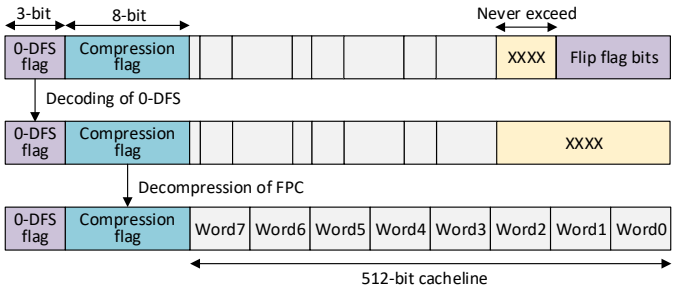



Fig. 13: Decoding procedure of CFS.

0-DFS for each cache line according to the saved space by compression.

Considering that cold data in the far segments of Tiered-crossbar arrays are not sensitive to the access performance, we propose *compression-based Flip Scheme* (*CFS*) in the far segments to reduce the sneak currents and leakage energy. CFS dynamically selects the most appropriate 0-DFS for each cache line according to the saved space by compression. Similar to CIDM, CFS is also implemented at the cache line granularity. CFS calculates the total saved space after the compression of the cache line, and then selects the 0-DFS that applies as small word size as possible on the condition that the total encoded data size will never exceed the original cache line size. Thus, the MSBs of the compressed cache lines in the far segments are encoded with different word-size 0-DFSs. Table V presents the most appropriate 0-DFS for each cache line according to the saved space by compression. A 3-bit 0-DFS flag for each cache line is used to denote the encoding method, e.g., '000' for the 2-bit word-size 0-DFS, '001' for the 4-bit word-size 0-DFS, '010' for the 8-bit word-size 0-DFS, '011' for the 16-bit word-size 0-DFS and '100' for no 0-DFS.

CFS consists of the CFS encoder module on the write path and the CFS decoder module on the read path. As shown in Figure 7, the CFS encoder and decoder modules are embedded inside the ReRAM controller as Path ②.

***Write Path.*** When the far segments receive a write request from the processor, the CFS encoder works as Figure 12 shows. The incoming cache line is first sent to the FPC compression logic to attempt data compression. The 8-word cache line is compressed word by word. The 8-bit compression flag is set to indicate whether each word is compressed or not. CFS contiguously stores the eight compressed words and then calculates the total saved space of the eight words. CFS also

TABLE VI: Simulation Configurations

| Parameter | Value |
|---|---|
| CPU | 4-Core, out of order, 3GHz, 192-entry recoder buffer, 8 issue width |
| L1 Cache | Private, 16KB I-cache, 16KB D-cache, 2-way assoc, 2-cycle access latency |
| L2 Cache | Private, 1MB, 64B cache line, 8-way assoc, 20-cycle access latency |
| L3 cache | Shared, 16MB, 64B cache line, 16-way assoc, 50-cycle access latency |
| Main memory | 8GB, DDR3-1333, 4 channel, 2 ranks/channel, 32 banks/rank, 1024 crossbar arrays/bank |
| ReRAM Timing(ns) | tRCD(18), tCL(15), tCWD(13), tFAW(30), tWTR(7.5), tWR(refer to Figure 9) |

reads out the MSBs of the compressed data. After that, CFS selects the most appropriate 0-DFS for the MSBs according to the total saved space by compression. For the sake of decoding, the flip flag bits are stored at the end of the cache line. The MSBs of the compressed cache lines are encoded with different 0-DFSs. The 3-bit 0-DFS flag is set to represent the encoding method. The uncompressible cache lines are directly sent to the write circuit without the compression and 0-DFS encoding.

***Read Path.*** When the far segments receive a read request from the processor, the CFS decoder works as Figure 13 depicts. First, the 512-bit cache line with the corresponding 0-DFS flag and compression flag is read out. Then the 512-bit cache line is decoded by the 0-DFS decoder module according to the 0-DFS flag and the flip flag bits. After that, the decoded cache line is decompressed word by word according to the compression flag and the prefix of each word.

## V. Experimental Methodologies

At the circuit level, we use our ReRAM circuit model to achieve the write latency and energy parameters of the near and far segments, as shown in Figure 9. We also use NVsim [32] to obtain the power, latency and area parameters of additional circuits. Then we add these parameters to our architectural simulator.

At the architecture level, we use GEM5 [33] with the integration of NVMain [34] as our simulator to evaluate the proposed techniques. The detailed simulation configurations are presented in Table VI. Most ReRAM-related memory timing parameters are obtained from the prior work [4]. The write latency (tWR) parameters of the near and far segments are derived from the circuit simulation (Refer to Figure 9). We select 10 benchmarks from SPEC CPU2006 with different memory Read Per Kilo Instructions (RPKI) and memory Write Per Kilo Instructions (WPKI) rates, as shown in Table VII. We run all the selected benchmarks for 500 million instructions to warm up caches and then run 1 billion instructions for the proposed techniques. We choose DSGB [4]+IDM((8,6),2) [10] as the aggressive baseline, which applies IDM((8,6),2) in DSGB-based crossbar arrays to reduce write latency/energy. The comparison configurations are listed as follows:

- *baseline*: Apply IDM((8,6),2) in DSGB-based crossbar arrays to reduce write latency/energy.

TABLE VII: RPKI and WPKI of SPEC CPU2006 Benchmarks

| Benchmark | Description | RPKI | WPKI |
|---|---|---|---|
| cactusADM | Four copies of cactusADM | 6.82 | 6.61 |
| astar | Four copies of astar | 2.04 | 1.05 |
| mcf | Four copies of mcf | 2.24 | 1.23 |
| bwaves | Four copies of bwaves | 10.14 | 9.62 |
| bzip2 | Four copies of bzip2 | 2.32 | 1.17 |
| wrf | Four copies of wrf | 8.18 | 7.88 |
| milc | Four copies of milc | 1.28 | 1.12 |
| gobmk | Four copies of gobmk | 1.65 | 1.44 |
| libquantum | Four copies of libquantum | 7.31 | 7.06 |
| sjeng | Four copies of sjeng | 8.32 | 8.13 |

- *Tiered-crossbar*: Apply the Tiered-crossbar design.
- *CIDM*: Apply the compression-based IDM technique in the whole crossbar array based on *Tiered-crossbar*.
- *Tiered-ReRAM*: Apply the compression-based IDM in the near segments and the compression-based flip scheme in the far segments based on *Tiered-crossbar*.

## VI. Experimental Results

### A. Overhead Analysis

**Additional circuit overheads.** We use NVsim [32] to evaluate the power, latency and area overheads of Tiered-ReRAM, which mainly come from the additional isolation transistors, encoders, decoders and multiplexers. The transistor device characteristics are derived from Narasimha's work [35] and scaled down to 22nm technology. The results show that the isolation transistors only incur $20.5pW$ power, $150ps$ latency and $0.37\%$ area overheads in a $512\times512$ crossbar array, which are acceptable. In the CIDM and CFS encoders, the latency of FPC compression is $2ns$ and the latency of IDM/0-DFS encoding is $1ns$. In the CIDM and CFS decoders, the latency of FPC decompression is $1ns$ and the latency of IDM/0-DFS decoding is smaller than $1ns$. FPC and IDM cost $10K$ gates in total, which only causes $0.1\%$ logic overhead. The multiplexers used to calculate the total saved space of the compressed cache line incur $1.75ns$ latency. The additional control logic to determine the near/far segments access only costs $280ps$ latency. Note that Tiered-ReRAM can look up the write latency table in parallel to the write operation. During the long-latency write operation, the memory controller can terminate the write operation according to the table lookup result. Therefore, the write latency table lookup doesn't incur latency overhead.

**Storage overhead.** In the CIDM technique, the additional 2-bit IDM flag is required to denote the selected IDM encoding method. The additional 8-bit compression flag is needed to indicate whether the eight words of a cache line are compressed or not. Therefore, the CIDM technique incurs 1.95% storage overhead in the near segments. In the CFS technique, the additional 3-bit 0-DFS flag is required to represent the selected 0-DFS encoding method. Similar to the CIDM technique, the additional 8-bit compression flag is also needed in the CFS technique. Thus, the CFS technique results in 2.15% storage overhead in the far segments. Besides, the write latency table has 16 entries and each entry requires 16 bits. Therefore, the table incurs $32B$ storage overhead for the memory controller.

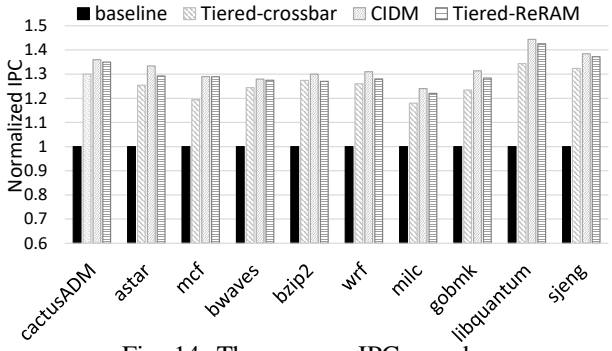



Fig. 14: The average IPC speedup.

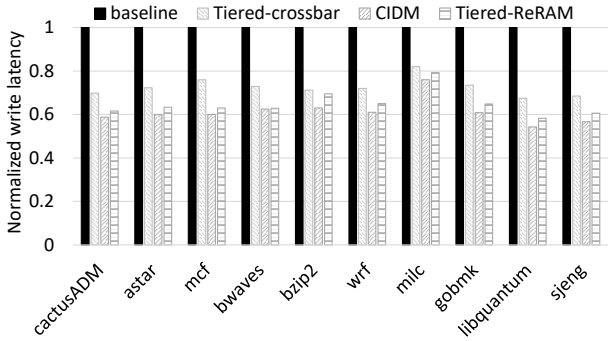



Fig. 15: The average memory write latency.

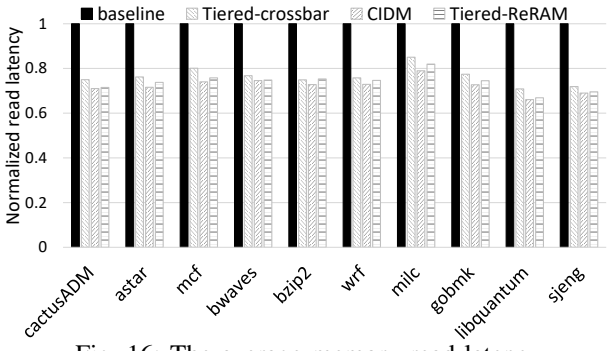



Fig. 16: The average memory read latency.

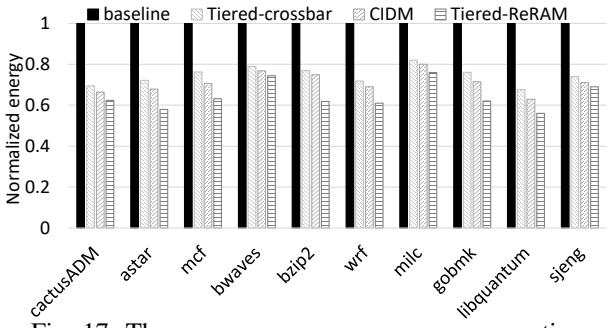



Fig. 17: The average memory energy consumption.

In addition, in order to remap hot data to the near segments and cold data to the far segments in the Tiered-crossbar design, the address remapping table is required in the memory controller and the table causes 256KB storage overhead for the 8GB TLC ReRAM.

### *B. System Performance*

We use the IPC (Instructions Per Cycle) speedup to evaluate the system performance. The IPC speedup is defined as follows.

$$IPC_{speedup} = \frac{IPC}{IPC_{baseline}}$$

Figure 14 shows the average IPC speedup of different design configurations with the results normalized to *baseline*. By applying our proposed techniques step by step, on average, *Tiered-crossbar*, *CIDM* and *Tiered-ReRAM* can improve the system performance by 26.1%, 32.6% and 30.6%, respectively. *CIDM* achieves the best performance because both the Tiered-crossbar design and the compression-based IDM technique in the whole crossbar array can significantly reduce the write latency. Compared to *CIDM*, *Tiered-ReRAM* has 2% fewer performance improvements because the CFS technique in *Tiered-ReRAM* programs more high resistance cells in the far segments and slightly increases the write latency. However, *Tiered-ReRAM* effectively reduces the sneak currents and leakage energy through these high resistance cells.

### *C. Write Latency*

Figure 15 illustrates the average write latency of different design configurations. The results are normalized to *baseline*. We can observe that our proposed techniques can significantly reduce the write latency. On average, *Tiered-ReRAM* achieves 35.2% write latency reduction over *baseline*. Compared to *baseline*, *Tiered-crossbar* can achieve 27.4% write latency reduction due to the effectiveness of the Tiered-crossbar design. Compared to *Tiered-ReRAM*, *CIDM* gets 3.5% more write latency reduction because the CFS technique in *Tiered-ReRAM* slightly increases the write latency of the far segments. However, *Tiered-ReRAM* can effectively decrease the sneak currents and leakage energy through the CFS technique.

### *D. Read Latency*

Since the Tiered-crossbar design and compression-based IDM technique can significantly reduce the write latency, the write service time is also reduced and read requests benefit from the waiting time reduction. Therefore, the overall read latency is also shortened. Figure 16 depicts the average read latency of different design configurations with the results normalized to *baseline*. The results show that, on average, *Tiered-ReRAM* decreases the read latency by 26.1% compared to *baseline*. Compared to *Tiered-crossbar*, *Tiered-ReRAM* gets 2.5% more read latency reduction due to the CIDM technique of the near segments in *Tiered-ReRAM*. On the other hand, due to the CFS technique of the far segments in *Tiered-ReRAM*, *CIDM* has 1.5% more read latency reduction than *Tiered-ReRAM*.

### *E. Energy Consumption*

The energy consumption in Tiered-ReRAM mainly comes from five sources: read operations, write operations, isolation transistors, encoders and decoders. Although the isolation transistors, encoders and decoders consume additional energy, Tiered-ReRAM can still decrease the energy consumption due

to the significantly reduced read and write latency. Moreover, the CFS technique in the far segments of Tiered-ReRAM effectively reduces the leakage energy.

Figure 17 illustrates the average energy consumption of different configurations with the results normalized to *baseline*. The results show that, on average, *Tiered-crossbar*, *CIDM* and *Tiered-ReRAM* reduce the energy consumption by 25.4%, 28.9% and 35.6%, respectively. Compared to *CIDM*, *Tiered-ReRAM* can achieve 6.7% more energy reduction. That's because the CFS technique in *Tiered-ReRAM* increases the number of high resistance ReRAM cells in crossbar arrays and significantly decreases the sneak currents and leakage energy.

## VII. Related Work

### A. Mitigating IR drop issue

The IR drop issue of crossbar arrays significantly increases the write latency and energy, impeding the development of ReRAM-based memory systems. Numerous works focus on mitigating the IR drop issue of crossbar arrays. Xu et al. [4] proposed Double Sided-Ground Biasing (DSGB) design to reduces the IR drops along wordlines by applying another ground on the other side of the selected wordline. However, DSGB fails to consider the IR drops of the long bitlines, resulting in significant performance degradation and energy waste. Different from DSGB design, the proposed Tiered-crossbar design splits each long bitline into the near and far segments by an isolation transistor, which fundamentally reduces the IR drops of the near segments. In addition, the Tiered-crossbar design only incurs 0.37% area overhead based on DSGB design. Zhao et al. [36] proposed the 1TnR V-ReRAM design to decrease the IR drops by changing the directions of access lines and reorganizing the peripheral circuitry. Shevgoor et al. [37] proposed a novel sample and hold circuit to mitigate the impact of IR drops on read operations. Zhang et al. [12] and Wen et al. [5] proposed to reduce the IR drops by optimizing the data patterns and writing more 0s into SLC crossbar arrays. Different from the these works, our proposed techniques aim to optimize the write operation of TLC crossbar arrays.

### B. Leveraging Non-uniform Access Latency

Since ReRAM cells at different locations of the crossbar array suffer from different IR drops, this characteristic causes non-uniform access latency in crossbar arrays. Conventional ReRAM writes use the worst-case access latency of all cells, resulting in significant performance degradation. Many works focus on leveraging the non-uniform access latency in crossbar arrays to improve the access performance. Based on the observation that the access latency of crossbar arrays is relevant to the distance between selected row and and write drivers, Zhang et al. [13] proposed the Leader design to partition each crossbar array into fast and slow regions by rows. Then static mapping and dynamic mapping methods were proposed to remap hot data to fast regions and cold data to slow regions, which significantly improves the access performance. In Tiered-crossbar design, we also adopt the dynamic mapping method. Compared with the Leader design, Tiered-crossbar fundamentally mitigates the IR drop issue of the near segments, providing much better performance for the near segments. Based on the observation that the access latency of crossbar arrays varies even in the same row, Zhang et al. [14] proposed a fine-grained region partition and address remapping scheme to further improve the access performance.

In other memory technologies, there are also numerous works leveraging the non-uniform access latency to improve performance. Son et al. [15] proposed to design some fast banks with shorter bitlines for faster data sensing and closer placement to the chip I/O for faster data transfers. Lee et al. [16] proposed to partition each DRAM subarray into fast and slow regions using the isolation transistors. Due to the irregularity in the manufacturing process, Chang et al. [17] exploited the access latency variation of DRAM subarrays. All these works allocated hot data to fast regions to improve the access performance.

### C. Incomplete Data Mapping

Incomplete Data Mapping (IDM) can effectively reduce the write latency and energy of TLC ReRAM. Based on the observation that programming different TLC states costs different latency and energy, Niu et al. [10] proposed IDM((8,6),2) method by eliminating two latency/energy critical states. In IDM((8,6),2), two 6-state cells are used to denote 5 digit bits because $log_2 6^2 \approx 5$. Although IDM((8,6),2) can reduce the write latency and energy of TLC ReRAM, it incurs 20% space overhead. Palangappa et al. [11] proposed to combine data compression techniques with the expansion coding (Similar to IDM) to reduce write latency and energy of TLC NVM. However, the space overhead of the expansion code is fixed and the saved space by compression can't be fully utilized. Different from the two techniques, the proposed CIDM technique dynamically selects the most appropriate IDM for each cache line according to the saved space by compression, which makes full use of the saved space by compression for more write latency/energy reduction.

### D. Flip Scheme

Generally, flip Schemes are used in memory technologies to reduce the bit flips. Flip-N-Write [38] and FlipMin [39] were proposed to reduce the bit flips of PCM writes. Flip-N-Write flips the new data if the number of different bits is more than half with comparison to the old data. FlipMin first uses the coset code to encode each possible input data vector into 256 different vectors, and then selects the vector with the minimum bit flips to write. Some other flip schemes can achieve specific effects. Zhang et al. [12] proposed 0-DFS to increase the number of 0s in SLC crossbar arrays for reducing the sneak currents. 0-DFS is a row-only flip scheme. 0-DFS flips the data with a flip flag bit '0' if the number of 0s in the data is smaller than or equal to N/2, where N is the word size of 0-DFS. CAFO [18] was originally proposed to reduce the write costs of the asymmetric memory. CAFO flips all the rows and columns that incur a positive gain from the cost model. CAFO can also be used to increase

the number of 0s. By simultaneously flipping the rows and columns, CAFO can achieve the most 0s. However, all the flip schemes cost high storage overhead due to the flip flag bits. Different from these previous works, the proposed CFS technique subtly combines data compression technique with the flip scheme to store the flip flag bits with the saved space by compression. CFS dynamically selects the most appropriate flip scheme for each cache line according to the saved space by compression and reduces the leakage energy with insignificant space overhead.

## VIII. Conclusion

In this paper, we propose Tiered-ReRAM architecture to reduce the write latency and energy of TLC crossbar ReRAM. Tiered-ReRAM is composed of Tiered-crossbar, CIDM and CFS designs. To reduce the IR drops in DSGB crossbar arrays, Tiered-crossbar splits each long bitline into the near and far segments by an isolation transistor. Tiered-crossbar allows the near segments to be accessed with decreased latency and energy. To further reduce the write latency and energy, CIDM is proposed in the near segments by dynamically selecting the most appropriate IDM for each cache line according to the saved space by compression. To ensure more high resistance cells written into crossbar arrays and reduce the leakage energy, CFS is proposed in the far segments by dynamically selecting the most appropriate flip scheme for each cache line according to the saved space by compression. The experimental results show that, Tiered-ReRAM can improve the system performance by 30.5%, reduce the write latency by 35.2%, decrease the read latency by 26.1%, and reduce the energy consumption by 35.6%, compared to an aggressive baseline.

## Acknowledgment

We thank the anonymous reviewers for their constructive comments and suggestions. We also thank anyone who helped us improve this work.

## References

[1] ITRS Roadmap. International technology roadmap for semiconductors. *Semiconductor Industry Association*, 2013.

[2] B. Lee et al. Architecting phase change memory as a scalable dram alternative. *ACM SIGARCH Computer Architecture News*, 2009.

[3] E. Kültürsay et al. Evaluating stt-ram as an energy-efficient main memory alternative. In *ISPASS*, 2013.

[4] C. Xu et al. Overcoming the challenges of crossbar resistive memory architectures. In *HPCA*. IEEE, 2015.

[5] W. Wen et al. Speeding up crossbar resistive memory by exploiting in-memory data patterns. In *ICCAD*. IEEE, 2017.

[6] C. Wang et al. Daws: Exploiting crossbar characteristics for improving write performance of high density resistive memory. In *ICCD*, 2017.

[7] C. Xu et al. Understanding the trade-offs in multi-level cell reram memory design. In *DAC*, 2013.

[8] Y. Deng et al. Rram crossbar array with cell selection device: A device and circuit interaction study. In *TED*, 2013.

[9] S. Sheu et al. A 4mb embedded slc resistive-ram macro with 7.2ns read-write random-access time and 160ns mlc-access capability. In *ISSCC*, 2011.

[10] D. Niu et al. Low power multi-level-cell resistive memory design with incomplete data mapping. In *ICCD*, 2013.

[11] P. Palangappa et al. Compex: Compression-expansion coding for energy, latency, and lifetime improvements in mlc/tlc nvm. In *HPCA*, pages 90–101, 2016.

[12] 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 TACO*, 2018.

[13] Hang Zhang et al. Leader: Accelerating reram-based main memory by leveraging access latency discrepancy in crossbar arrays. In *DATE*, 2016.

[14] Y. Zhang et al. Asymmetric-reram: A low latency and high reliability crossbar resistive memory architecture. In *ISPA*, 2018.

[15] Y. Son et al. Reducing memory access latency with asymmetric dram bank organizations. *ACM SIGARCH Computer Architecture News*, 41(3):380–391, 2013.

[16] D. Lee et al. Tiered-latency dram: A low latency and low cost dram architecture. In *HPCA*, pages 615–626, 2013.

[17] K. Chang et al. Understanding latency variation in modern dram chips: Experimental characterization, analysis, and optimization. *ACM SIGMETRICS Performance Evaluation Review*, 44(1):323–336, 2016.

[18] R. Maddah et al. Cafo: Cost aware flip optimization for asymmetric memories. In *HPCA*, pages 320–330, 2015.

[19] HY Lee et al. Evidence and solution of over-reset problem for hfox based resistive memory with sub-ns switching speed and high endurance. In *IEDM*. IEEE, 2010.

[20] M. Lastras-Montaño et al. A low-power hybrid reconfigurable architecture for resistive random-access memories. In *HPCA*, pages 102–113, 2016.

[21] Gennady Pekhimenko et al. Linearly compressed pages: a low-complexity, low-latency main memory compression framework. In *MICRO*, pages 172–184. ACM, 2013.

[22] Gennady Pekhimenko et al. A case for toggle-aware compression for gpu systems. In *HPCA*, pages 188–200. IEEE, 2016.

[23] Jungrae Kim et al. Bit-plane compression: Transforming data for better compression in many-core architectures. In *ISCA*, pages 329–340. IEEE, 2016.

[24] Jayesh Gaur, Alaa R Alameldeen, and Sreenivas Subramoney. Base-victim compression: an opportunistic cache compression architecture. In *ISCA*, pages 317–328. IEEE, 2016.

[25] A. Alameldeen et al. Frequent pattern compression: A significance-based compression scheme for l2 caches. In *Dept. Comp. Scie., Univ. Wisconsin-Madison, Tech. Rep,*, 2004.

[26] G. Pekhimenko et al. Base-delta-immediate compression: Practical data compression for on-chip caches. In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, 2012.

[27] Y. Zhang et al. A novel reram-based main memory structure for optimizing access latency and reliability. In *DAC*. ACM, 2017.

[28] Burr et al. Large-scale (512kbit) integration of multilayer-ready access-devices based on mixed-ionic-electronic-conduction (miec) at 100% yield. In *VLSIT*, 2012.

[29] Yoongu Kim et al. A case for exploiting subarray-level parallelism (salp) in dram. *ACM SIGARCH Computer Architecture News*, 40(3):368–379, 2012.

[30] Donghyuk Lee et al. Adaptive-latency dram: Optimizing dram timing for the common-case. In *HPCA*, pages 489–501. IEEE, 2015.

[31] Donghyuk Lee et al. Design-induced latency variation in modern dram chips: Characterization, analysis, and latency reduction mechanisms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):26, 2017.

[32] X. Dong et al. Nvsim: A circuit-level performance, energy, and area model for emerging non-volatile memory. *TCAD*, 2014.

[33] Nathan et al. The gem5 simulator. *SIGARCH Comput.Archi.News*, 2011.

[34] Matt Poremba et al. Nvmain: An architectural-level main memory simulator for emerging non-volatile memories. In *ISVLSI*. IEEE, 2012.

[35] S. Narasimha et al. High performance 45-nm soi technology with enhanced strain, porous low-k beol, and immersion lithography. In *IEDM*, 2006.

[36] L. Zhao et al. Constructing fast and energy efficient 1tnr based reram crossbar memory. In *ISQED*, 2017.

[37] M. Shevgoor et al. Improving memristor memory with sneak current sharing. In *ICCD*, pages 549–556, 2015.

[38] S. Cho et al. Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance. In *MICRO*, pages 347–357, 2009.

[39] A. Jacobvitz et al. Coset coding to extend the lifetime of memory. In *HPCA*, 2013.