Overcoming the Challenges of Crossbar Resistive Memory Architectures

Cong Xu†, Dimin Niu†, Naveen Muralimanohar¶, Rajeev Balasubramonian♯¶,
Tao Zhang†, Shimeng Yu‡, Yuan Xie§†
†Pennsylvania State University {czx102,dun118,tzz106}@cse.psu.edu
¶HP Labs naveen.muralimanohar@hp.com ♯University of Utah rajeev@cs.utah.edu
‡Arizona State University shimeng.yu@asu.edu
§University of California Santa Barbara yuanxie@ece.ucsb.edu

Abstract

The scalability of DRAM faces challenges from increasing power consumption and the difficulty of building high aspect ratio capacitors. Consequently, emerging memory technologies including Phase Change Memory (PCM), Spin-Transfer Torque RAM (STT-RAM), and Resistive RAM (ReRAM) are being actively pursued as replacements for DRAM memory. Among these candidates, ReRAM has superior characteristics such as high density, low write energy, and high endurance, making it a very attractive cost-efficient alternative to DRAM.

In this paper, we present a comprehensive study of ReRAM-based memory systems. ReRAM's high density comes from its unique crossbar architecture where some peripheral circuits are laid below multiple layers of ReRAM cells. A crossbar architecture introduces special constraints on operating voltages, write latency, and array size. The access latency of a crossbar is a function of the data patterns involved in a write operation. These combined with ReRAM's exponential relationship between its write voltage and switching latency provide opportunities for architectural optimizations. This paper makes several key contributions. First, we study the crossbar architecture and describe trade-offs involving voltage drop, write latency, and data pattern. We then analyze microarchitectural enhancements such as double-sided ground biasing and multi-phase reset operations to improve write performance. At the architecture level, a simple compression based data encoding scheme is proposed to further bring down the latency. As the compressibility of a block varies based on its content, write latency is not uniform across blocks. To mitigate the impact of slow writes on performance, we propose and evaluate a novel scheduling policy that makes writing decisions based on latency and activity of a bank. The experimental results show that our architecture improves the performance of a system using ReRAM-based main memory by about 44% over a conservative baseline and 14% over an aggressive baseline on average, and has less than 10% performance degradation compared to an ideal DRAM-only system.

1. Introduction

DRAM has been used in main memory for more than four decades. However, recent technological trends seriously challenge the continued dominance of DRAM, and open up new possibilities for future main memory systems. Consider the following trends that shape the landscape of future memories:

- *Scalability:* DRAM capacitors have inherent limitations for continued cell scaling. In order to ensure enough noise margin, the capacitance of a DRAM cell must be maintained at a relatively constant value regardless of technology scaling. Since the cell capacitance is directly related to the surface area of the capacitor, as we shrink the cell size, we have to compensate the decrease in width by increasing the height of the capacitor. As a result, the aspect ratio of both trench capacitor and stack capacitor increase rapidly with technology scaling. Due to limitations in the manufacturing process, the scaling path of DRAM beyond 16 *nm* is not clear according to the ITRS [15].
- *Memory Density:* The amount of data we process is rapidly increasing at a rate higher than that of Moore's law. To meet this growing demand, in addition to conventional scaling, industry is exploring other options such as multi-level cell (MLC), 3D stacking, and multi-layer structure to improve memory density. Therefore, emerging technologies with these benefits, such as PCM and ReRAM, are becoming increasingly attractive for future systems.
- *Main Memory as Primary Data Store:* Due to the growing performance gap between main memory and storage, many emerging workloads have begun to use main memory as the primary data store. Memcached, SAP HANA, and VoltDB are some examples of these workloads. Researchers have been exploring novel architectures around this idea such as a flat hierarchy with combined memory and storage [38], and byte addressable persistent memory for high performance [7]. With these, there is a demand for not just large capacity, but also for memories with high availability and large retention time.
- *Refresh Overhead:* As the DRAM memory capacity keeps increasing, static overhead due to *refresh* operations becomes non-trivial [45, 23, 56]. Refresh operations both increase power consumption, and hurt memory performance. In addition, the refresh cycle time tRFC increases with the capacity of the DRAM device. For example, the tRFC of a 4Gb DDR3 DRAM device is about 260*ns* and that of a 16 Gb DRAM device is predicted to exceed 1μs [23]. For future large memory systems, technologies with low or zero static power overhead are preferred over DRAM.

In the past few years, many researchers have explored PCM technology for main memory – both as a stand-alone memory or in conjunction with DRAM [19, 37, 57]. While PCM has many favorable characteristics, there are a few critical drawbacks that preclude PCM from emerging as a clear winner.

This work is supported in part by NSF 1213052, 1461698, 1500848 and by the Department of Energy under Award Number DE - SC0005026.

Writing a PCM cell is slow ($\sim 300ns$) and consumes high energy ($\sim 30pJ$) [19], resulting in low write bandwidth. A 20nm 8Gb PCM prototype only provides tens of MB/s write bandwidth [6]. Multi-level-cell (MLC) PCM further increases the write latency and energy by another order of magnitude. In addition, MLC PCM suffers from long-term and short-term resistance drift [3]. In fact, a recent study has pointed out that it may not be practical to use a 2b/cell MLC PCM as main memory [42].

ReRAM is another promising non-volatile memory technology that shares several positive characteristics of PCM. Unlike PCM, it does not have the resistance drift problem, and it has relatively lower write energy and higher density. While some recent prototypes of ReRAM are optimized for density trading off latency (e.g., a 32 Gb part from SanDisk [24] targeted as flash replacement), the technology can also be leveraged to build low latency memories. A recent prototype from Micron [9] and HP's Memristor crossbar project [13] are the best examples of industry efforts to leverage latency-optimized ReRAM for main memory. ReRAM has demonstrated superior endurance ($> 10^{10}$ [20, 22]), significantly alleviating the wear-out problem in PCM. In addition, it can benefit from the solutions proposed for the PCM [5, 16, 16, 37, 57, 36, 40, 14, 16, 35, 39, 41, 53, 1].

In this paper, we analyze design challenges for an ReRAM-based memory architecture, and propose circuit-level and architecture-level optimizations to enable the adoption of this emerging technology for future memory systems design. We make several key contributions. First, we study the crossbar architecture and describe trade-offs involving voltage drop, write latency, and data pattern for both a conservative baseline design and an aggressive design with double-sided ground biasing. Second, we split the long-latency RESET operations to sub-phases (hRESET) to further reduce write latency. Third, we propose a simple compression based encoding scheme with negligible storage overhead to speed up most of the write operations by limiting the worst-case voltage drop across the selected cells. Finally, we present and evaluate a memory scheduling policy that considers the varying latency of ReRAM writes along with pending activity of a bank when flushing writes to the memory. In addition to improving performance, the proposed design can lower energy, and has no first-order effect on endurance.

While prior work has architected PCM designs that can be used effectively as main memory, similar strategies do not apply to ReRAM cells. The key difference is that ReRAMs are best implemented with a dense crossbar architecture, while most architecture-level optimizations for PCM typically assume that a PCM cell uses either a MOSFET or a bipolar junction transistor (BJT) as an access transistor [2]. Therefore, a different set of innovations is required and this paper describes these crossbar-specific innovations. To our knowledge, this is the first architecture paper that shows the significance of sneak current in a crossbar architecture and its impact on latency, especially for bipolar ReRAM. In addition, the compression based encoding, its effect on write latency, and the proposed scheduling techniques are novel additions to the non-volatile main memory literature.

[2] While PCM can also use a crossbar, PCM being a unipolar device with different SET and ReSET latencies, its cell diode places very different constraints and has lower sneak current compared to ReRAM.

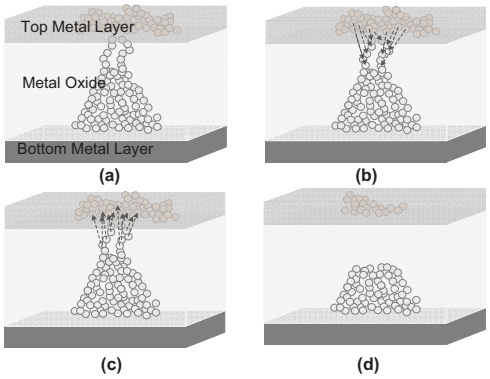

Figure 1: (a) ReRAM cell with conductive filament (b) RESET operation (c) SET operation (d) HRS of an ReRAM

2. Resistive Memory Technology

Resistive memories, in general, refer to any technology that uses varying cell resistance to store information. However, the moniker ReRAM typically refers to the subset that use metal oxides as the storage medium, called referred to as metal-oxide ReRAM.

The schematic view of a metal-oxide ReRAM cell is shown in Figure 1a. It has a very simple structure: a metal-oxide layer is sandwiched between two layers of metal electrodes, named top electrode and bottom electrode. Similar to PCM, a low resistance state (LRS or ON-state) and a high resistance state (HRS or OFF-state) are used to represent the logical "1" and "0" respectively. In order to switch an ReRAM cell, an external voltage with specific polarity, magnitude, and duration is applied to the sandwiched layer. The switching of LRS-to-HRS is called a RESET operation and the switching of HRS-to-LRS is called a SET operation.

The filamentary model has been widely accepted in explaining the switching process of ReRAM [51], which attributes state changes to the formation and the rupture of nanoscale conductive filaments (CFs) within a cell. The SET and RESET operations are shown in Figure 1b and Figure 1c. During a RESET, the oxygen ions are forced back to the oxide layer by the electric field and they recombine with the oxygen vacancies. In this case, the CFs are "cut off" and the cell transitions to the high resistance state, which is shown in Figure 1d. A SET operation (Figure 1c), which switches the cell back to the low resistance state, realizes the regeneration of the CFs by drifting the oxygen ions to the anode layer and leaving the oxygen vacancies in the metal oxide layer (Figure 1a).

Tens of binary metal oxide materials have demonstrated resistive switching behavior under an electric field [51] and different ReRAM materials have wildly different characteristics. For this work, we will focus on HfOx-based ReRAM technology owing primarily to its excellent scalability ($< 10nm$) [10],

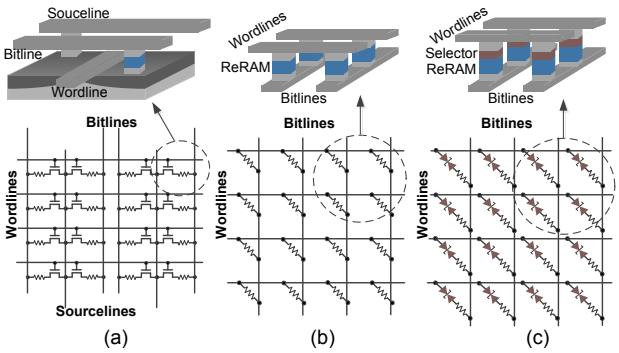

Figure 2: Overview of ReRAM array structures: (a) MOSFET-accessed structure; (b) access-device-free crossbar structure; (c) diode-accessed crossbar structure

superior switching speed/energy [10, 20], and high endurance ($> 10^{10}$) [20].

3. ReRAM Array Architecture

Based on the characteristics of ReRAM cells and energy/delay/cost constraints of the main memory, an ReRAM array can either be designed as a grid with 1T1R cells or as a dense *crossbar architecture*.

3.1. 1T1R Grid Architecture

Figure 2a shows a conventional design where each cell has a dedicated MOSFET transistor ("1T1R" structure). Similar to DRAM, when a row gets activated, the access transistors in the selected row provide exclusive access to the cells in that row without disturbing other cells in the array. Thus, it has a similar concept of row buffer, and the fetch width of an array depends on the size of the array. However, unlike DRAM, resistive memories typically operate at a significantly higher current, requiring a large sized access transistor for each cell. The size of these transistors ultimately increases the area and hence the cost. However, due to perfect isolation provided by these access transistors, the "1T1R" design is more energy efficient and has superior access time compared to other alternatives.

3.2. Crossbar Architecture

Figures 2b & 2c show a crossbar architecture, in which all cells are interconnected to each other without transistors. ReRAM cells are directly sandwiched between top and bottom electrodes. By eliminating access transistors, cells in a crossbar achieve the smallest theoretical size of $4F^2$[3]. Furthermore, as ReRAM cells employ different fabrication steps from transistors, the silicon area under the array can be used for other peripheral circuits such as decoders and drivers, maximizing the area efficiency of an array. In a highly cost conscious memory market, the crossbar architecture is better suited for ReRAM-based main memory. However, crossbar architectures introduce other challenges and this paper is an attempt to overcome some of those challenges.

[3] A cell size of $4F^2$ is literally area under the cross-section of a minimum sized wordline and biltine.

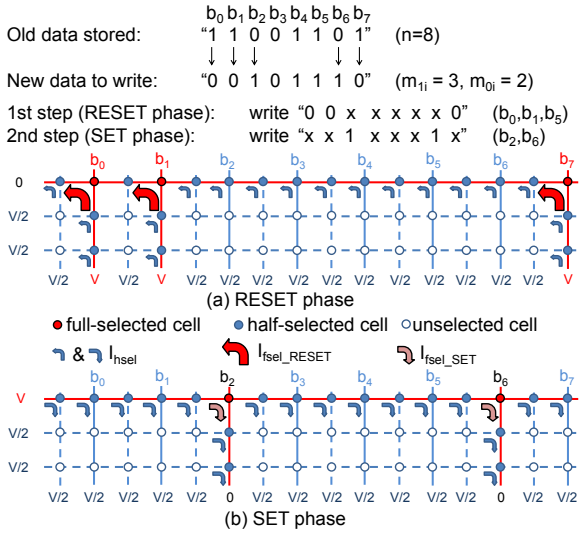

Figure 3: Two phase multi-bit write operation in a crossbar array: (a) RESET phase and (b) SET phase

In a crossbar design, it is possible to access a single cell in an array by applying the proper potential across the wordline and bitline to which the cell is connected. Hence, there is no over-fetch problem as in DRAM [49]. However, as selected cells are no longer isolated from unselected cells, activating a wordline and a bitline will result in current flow across all the cells in the selected row and column.

Ideally, when we activate a wordline and bitline(s), we want the entire current to flow through the *full-selected cell(s)* that lies at their intersection(s). For example, in order to SET specific cell(s) in the crossbar array, the selected wordline and bitline(s) are set to V (or V_W) and 0 respectively, as shown in Figure 3b. Therefore, the write voltage V is fully applied across the *full-selected cell(s)*. However, other cells in the selected row and column(s) also see partial voltage across them. These *half-selected cells* in the selected row and column leak current through them due to the partial write voltage across them, which is commonly referred to as *sneak current*. In order to reduce the sneak current, all of the other wordlines and bitlines that are not selected are half biased at $V/2$. This limits the voltage drop on the *half-selected cells* to $V/2$ and voltage drop on the *unselected cells* to 0. Note that a dashed line in Figure 3 represents more than one unselected wordlines or bitlines. ReRAM cells exhibit a non-linear relationship between their applied voltage and their current, i.e., current decreases significantly with a small drop in voltage. This helps keep the sneak current through half-selected cells in check. Thus, a critical parameter in a crossbar architecture is the ratio of the amount of current flowing through a *fully-selected cell* (I_{fsel_RESET}) to a *half-selected cell* (I_{hsel}), referred to as *non-linearity* (κ). The higher the κ, the lower the sneak current, and the higher the feasibility of a large crossbar array.

To be cost competitive with DRAM, a memory with fairly large mat size is necessary to reduce peripheral circuit overhead, and for this, κ of simple ReRAM alone is not sufficient [29]. Many recent ReRAM prototypes employ a ded-

icated selector or bi-polar diode in each cell to improve κ, as shown in Figure 2c [18, 21, 24]. Since a selector can be built on top of the switching material, there is no extra area overhead required for the selector. A selector can be built with many different materials with different endurance, operating voltage, and current densities. In this work, we model a selector similar to the one showcased by Burr et al. [4].

3.3. Constraints in Crossbar Architectures

Although a crossbar architecture is best suited for building dense memories, most ReRAM memories, even with a dedicated selector in each cell, have only finite non-linearity. Hence, irrespective of how good the cells are, the sneak current flowing through the cells poses a number of challenges, opening up new research possibilities for architects.

In a crossbar, the amount of sneak current ultimately determines the energy efficiency, access time, and area of an array. For example, ideally we want to build a big array to improve the density. However, as we size up an array, the number of *half-selected cells* increases. Thus we need to provide sufficient voltage at the driver to account for these sneak currents to avoid *write failure* [29]. However, high voltage at the driver should not be significant enough to disturb cells closer to the driver, which can lead to *write disturbance* [29]. Furthermore, since sneak current can vary based on the data stored in the array, it is critical to architect the sensing circuit and the array dimensions so that we have enough noise margin to differentiate sneak current from the total read current.

A critical characteristic of an ReRAM cell is that its switching time is inversely exponentially related to the voltage applied on the cell [10, 54]. The write latency of the furthest selected cell is calculated based on the relationship between its voltage drop V_d and switching time τ: $\tau \times e^{kV_d} = C$, where k and C are fitting constants extracted from experimental results [20]. For example, a HfOx-based ReRAM has demonstrated that a 0.4V reduction in RESET voltage may increase RESET latency by 10X [10].

Even though the switching time of some ReRAM cells can be small if they are located near the write driver and have almost full write voltage, many ReRAM cells in the mat will see a different voltage drop across the cell due to the IR drop introduced by the sneak current and the wire resistance. The current passing through the metal wires causes significant voltage loss on the metal wires and thus decreases the voltage drop on the furthest cell in an ReRAM crossbar. Therefore, the worst-case switching time of an ReRAM crossbar depends on the array size, write current, metal resistance, and number of bits being written in parallel in a row (wordline). A straightforward solution is to increase the output voltage of the write driver so that the worst-case voltage drop can be improved. However, this has several drawbacks: (1) higher voltage results in higher write power; (2) larger output voltage requirement increases charge pump stages and complexity, and thus corresponding area and energy overhead; (3) a larger $V/2$ increases the probability of *write disturbance*; (4) higher voltage introduces reliability issues, such as time-dependent gate oxide breakdown of the write driver and worse drain-induced barrier lowering effect. To deal with such issues, specialized transistor design is required, increasing both cost and area overhead; (5) the excessive voltage may over-RESET the nearest selected cells and cause stuck-at-0 faults, resulting in endurance degradation [20]. In this work, any optimization technique we introduce aims at increasing the worst-case (minimum) voltage drop on the selected cell without affecting the maximum voltage drop on any cell. Therefore, they should not further bring reliability issues and endurance degradation.

Table 1: Parameters in the Crossbar Array Model

Metric	Description	Value or Range
A	Mat size: A wordlines $\times A$ bitlines	$128 \sim 1024$
n	Number of bits to read/write	$1 \sim 64$
I_{on}	Cell current of a LRS ReRAM during RESET	$20uA$
R_{wire}	Wire resistance between adjacent cells	2.82Ω
K_r	Nonlinearity of the selector	3000
V_W	Full selected voltage during write	$3.2V$
V_R	Read voltage	$1.6V$

4. Modeling ReRAM

As we pointed out earlier, in emerging memory technologies such as ReRAM and PCM, the latency and energy of memory are primarily determined by cell characteristics and the mat architecture [4], whereas in DRAM the overhead of routing data to a mat is the dominating factor. To investigate how sneak current impacts the overall latency, energy, and area, we present a novel modeling tool to investigate crossbar and bank architectures. The tool is used to analyze the design space and isolate some feasible design points that will be optimized in the rest of the paper.

4.1. Array-Level Model

Many design considerations such as the mat size, operating voltage, driving/sensing circuit, biasing voltage, and operating current, that are fairly straightforward in existing memory technologies, require complex analysis for crossbar arrays.

To estimate the wordline current, voltage drop, switching time, and write power of a crossbar array, we develop a detailed crossbar array model. Wire resistance is considered in the model, and a full HSPICE netlist is generated for a diode-accessed crossbar array once the mat size (A) and number of bits to read/write in a mat (n) are given. The HSPICE model uses a combination of a selector and a resistive element to form a non-linear cell. We extract the device parameters from HfOx-based cells [20] and selector parameters from IBM's MIEC device [4]. Most important parameters in the model are summarized in Table 1. All of our analysis assumes SLC ReRAM with one layer and $4F^2$ cell size (smaller ReRAM cell size is feasible with multiple layers or multi-level cells).

To calculate the worst-case voltage drop on the selected cell(s), it is important to model how the writes are actually done. Typically a multi-bit write operation in a crossbar array is done in two phases: a RESET phase and a SET phase [52]. Figure 3 illustrates an 8-bit write in a crossbar array. After

[4] We refer to the grid of memory cells, the smallest building block of memory, as mat. In ReRAM, a mat is also referred to as a crossbar.

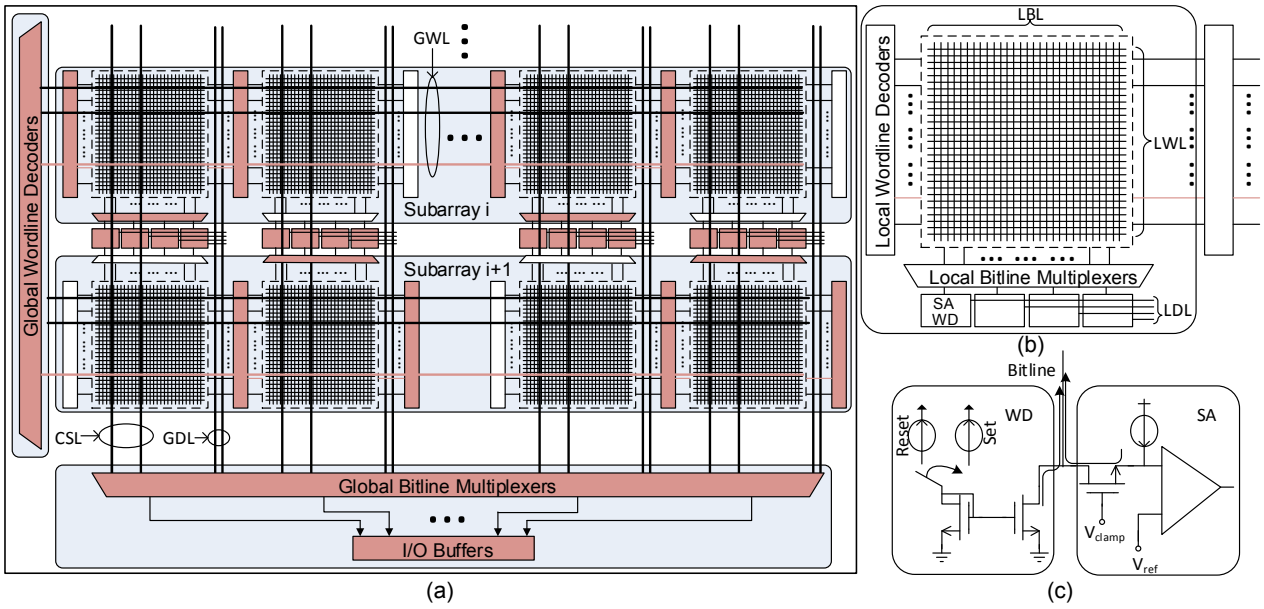

Figure 4: Modeling of ReRAM architecture in modified CACTI: (a) Schematic view of an ReRAM bank; (b) Mat organization; (c) Design of sense amplifier and write driver (CSL: column select line; GWL: global wordline; GDL: global dataline; LBL: local bitline; LWL: local wordline; LDL: local dataline; SA: sense amplifier; WD: write driver)

comparing the new 8-bit data with the old 8-bit data, bits b_0, b_1, b_7 need to be switched from "1" to "0" (we refer this number as m_{1i}) and b_2, b_6 need to be switched from "0" to "1" (we refer this number as m_{0i}). In the RESET phase, the three cells in the selected wordline will be fully selected. Note that every dashed horizontal (vertical) line in the figure represents more than one unselected wordline (bitline) for a large array size. Since all the selected cells in the RESET phase are in low resistance state and contribute to the total current on the selected wordline, the voltage drop on the furthest selected cell is significantly smaller than the output voltage from the write driver. As a result, the switching time of the furthest cell is much longer than the minimum RESET latency. This eventually becomes a performance bottleneck when many bits are mapped to a large crossbar array.

4.2. Bank-Level Model

After evaluating the current, latency, and power of a crossbar array using HSPICE, we integrated the array model into CACTI. We heavily modified CACTI to model the bank architecture of ReRAM illustrated in Figure 4. We changed the interface of CACTI so that it takes the HSPICE result for the crossbar and outputs the area/latency/power breakdowns of the subarray, bank, local/global wordline decoders, local/global bitline decoders, sense amplifiers/latches, and output drivers. Figure 4c shows the sense-amplifier and write driver being employed for ReRAM. We also modified the area model such that part of last level wordline/bitline drivers are laid underneath the crossbar arrays [18], which improves the overall area efficiency of the ReRAM chip.

The parasitic latency is based on HSPICE transient analysis and peripheral delay is estimated in CACTI. Some demonstrated ReRAM prototypes have microsecond-level latency [24] because they target the flash market. Their interface, limited sense amplifiers, large array size, and low circuit footprint are targeted for extremely low cost-per-bit, penalizing latency. We model an LPDDR-/DDR- compatible interface, and the circuit design decisions are to improve latency at the expense of density.

4.3. Design Constraints Analysis

In this Section, we study the impact of optimizing cost and power for a 22nm 8Gb ReRAM chip. The goal of this analysis is to identify a few reasonable baseline design points for the rest of the paper.

We assume a x8 ReRAM chip architecture with a DDR3-compatible interface and 64-bit internal prefetch width. These 64 bits are distributed to $64/n$ mats in one of the banks. Each activated mat reads (or writes) an n-bit group. Here n is also the minimum number of local sense amplifiers and write drivers in a mat.[5] Increasing n means fetching the 64 bits from fewer mats and each activated mat accessing more bits. We evaluate the area and bank write power of the ReRAM chip with various values of n and mat sizes. We assume that the number of wordlines is equal to the number of bitlines in a mat (denoted as A in Figure 5). Later in this paper, "A512n8" refers to the design point that has a mat with 512 wordlines and bitlines, and each mat deals with an 8-bit group.

Chip area is the key indicator of chip cost and a primary optimization parameter for the memory industry. Figure 5a shows that the chip area is reduced as the mat size increases.

[5] Actually each mat owns half the number of its local sense amplifiers and write drivers because they are shared among two adjacent mats, as illustrated in Figure 4a

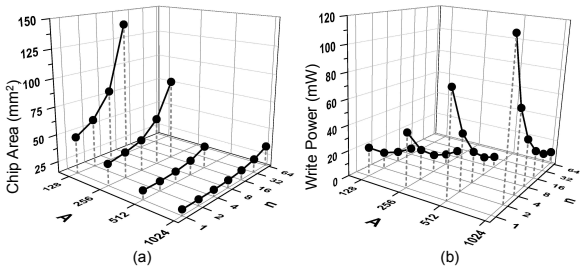

Figure 5: (a) Chip area and (b) Bank Write power of a 22nm 8Gb ReRAM chip with various mat sizes and number of bits to read/write per mat

Due to the large footprint of local sense amplifiers and write drivers, the chip area increases significantly as n increases. The increasing trend is more remarkable with a smaller mat size. In our work, the area constraint of an ReRAM chip is defined to be the area of a DRAM chip with the same capacity, and a 22nm 8Gb DRAM chip area is estimated to be $45mm^2$ assuming an array efficiency of 55% [15].

Write power contributes to a significant portion of the total power consumption in ReRAM. It is a critical design concern for any memory when power budget is limited, and in this work we apply the same power budget of DRAM to our ReRAM design. The write power of DRAM is calculated from a DRAM power calculator by using the datasheet from Micron [27]. Figure 5b illustrates that the write power of ReRAM goes down quickly as n increases. The reason is that the number of activated mats during write access is halved if n doubles, which leads to much less energy consumption due to sneak currents. In addition, a larger mat size increases the mat power with the increasing of the sneak current. To summarize, a larger n is preferred from a power perspective.

After applying the same chip area and power constraints of DRAM to ReRAM, a few ReRAM organizations meet the design criteria. The mat sizes of these organizations ranges from 256×256 to 1024×1024 while the values of n lie between 4, 8 and 16.

5. Architecting High Performance ReRAM

One of the biggest challenges in a crossbar architecture is to overcome the sneak current induced voltage loss. This problem is particularly challenging for ReRAM as the switching time of ReRAM varies exponentially with voltage drop across the cell. For a given mat size, the worst-case voltage drop across the *full-selected cell(s)* depends on two key parameters: 1) the biasing of unselected bitlines and wordlines; 2) the number of low resistance states in the selected row and column, i.e., the data pattern in the row and column. To understand how much biasing can help, we first discuss a microarchitectural enhancement called *double-sided ground biasing (DSGB)*. DSGB incurs an area penalty and might not be suitable in a cost conscious design. We simply use this as a second baseline to show the effectiveness of managing data patterns in a mat.

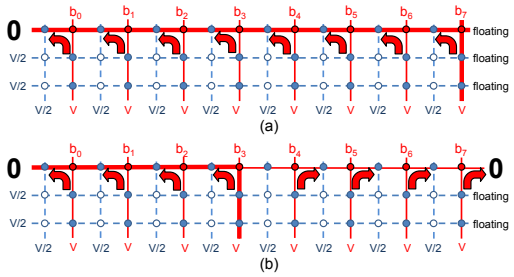

Figure 6: (a) The conventional biasing scheme during RESET phase, (b) the proposed double-sided ground biasing during RESET phase

5.1. Double-Sided Ground Biasing

The conceptual view of DSGB is demonstrated in Figure 6. Using "A512n8" as an example, the worst-case voltage drop occurs at cell b_7 which corresponds to the longest IR drop path (marked as the red bold line) from the write driver to the ground. In the conventional biasing scheme, as seen in Figure 6a, the ground is located at one side of the selected wordline during the RESET phase. The key idea of DSGB is to apply another ground on the other side of the selected wordline. By doing so, the length of the worst-case IR loss path has been reduced, as illustrated in Figure 6b. With DSGB, the worst-case voltage drop occurs at cell b_3, with significantly larger voltage than that observed on cell b_7 in Figure 6a. The results show that the worst-case voltage drop in "A512n8" has been improved from 2.146V to 2.328V. As a result, the RESET latency has been reduced from 682ns to 240ns.

Although this looks fairly straightforward, the design overhead could be significant because it requires either (a) an additional set of row decoders/drivers on the other side of each array with reduced area efficiency, or (b) shortening the two ends of the array using another metal layer with extra cost overhead. To minimize the area or routing overhead, we borrow the decoder signal from the adjacent array, as shown in Figure 4a & 4b. In this design, the opposite side of the unselected wordlines are still left floating, as illustrated in Figure 6b, and thus the overhead is limited topass transistor per row.

The problem of DSGB is the reduced data parallelism: only half of the mats can be activated within an selected subarray due to resource contention (local row decoders). Consequently, only half of the write drivers and sense amplifiers are utilized during each access. To tackle the issue, we present a simple yet effective solution with small modifications of the ReRAM bank architecture. As in the conventional design, the write drivers and sense amplifiers are already shared between adjacent subarrays. It is possible to utilize these resources in alternating fashion. In the proposed design, as illustrated in Figure 4a, each time a pair of adjacent subarrays are selected in a bank. Subarray i activates odd-numbered mats while subbarray $i+1$ activates even-numbered mats. Therefore, the data parallelism has been maintained.

In summary, DSGB can improve access speed by increasing the voltage across the selected cell, but the downside is that it

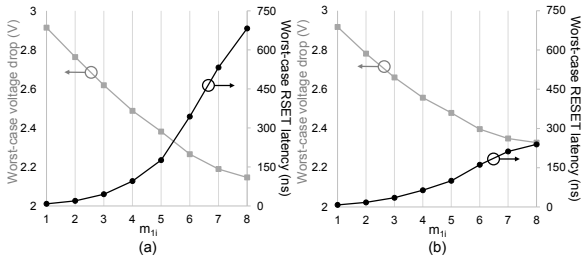

Figure 7: Impact of m_{1i} on the RESET latency of A512n8 for (a) baseline1 without DSGB, (b) baseline2 with DSGB

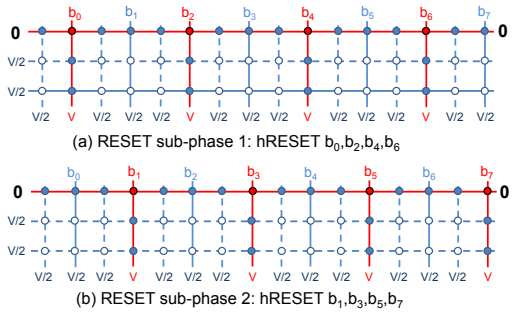

Figure 8: The illustration of split RESET phase under a worst case

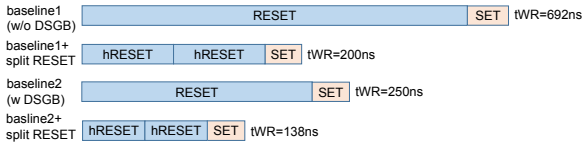

Figure 9: Reduction of write latency by DSGB and split RESET

incurs area overhead. As we mentioned earlier, we describe DSGB not only to show the effectiveness of this technique, but also to understand the extent to which the data pattern impacts the write latency. As we show in the next sub-section, in spite of improving biasing with enhancements such as DSGB, there is still significant voltage loss due to the sneak current.

5.2. Impact of Data Pattern on Write Latency

It is shown in Section 4.1 that the write latency for writing an n-bit group in a mat is a function of the number of "1" to "0" transitions (m_{1i}). The RESET phase can become a performance bottleneck for large m_{1i}, whereas the SET phase can always be completed in less than $10ns$ even if $m_{0i} = n$. To quantify the impact of m_{1i}, we evaluate the worst-case RESET latency of writing an 8-bit group in a 512×512 mat in a conservative baseline without DSGB (baseline1) and an aggressive baseline with DSGB (baseline2). From Figure 7, we can see that the worst-case RESET latency increases substantially as m_{1i} goes up for both cases. In the following subsections, we discuss techniques that either try to avoid worst-case data patterns or mitigate their impact on write latency.

5.2.1. Split RESET phase Due to super-linear relationship between m_{1i} and RESET latency in a crossbar, it is more efficient to write one bit at a time sequentially to a mat rather than to perform a multi-bit write operation to reduce tWR. However, this increases the bank decoding latency and the

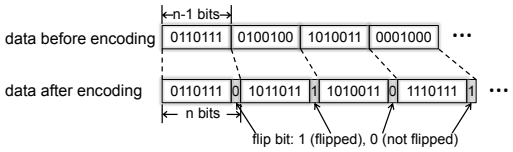

Figure 10: A (n-1,n) encoding demonstration

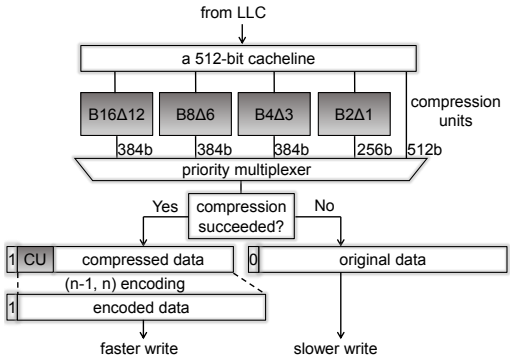

Figure 11: An overview of the compression/encoding flow

global bitline latency to send bits sequentially from memory IO pads to the write drivers of the subarray, which we refer to as tCL. Alternatively, we can have logic to perform sequential writes near the subarray but this increases silicon area per subarray and hence cost. We found that when write width per crossbar is eight, writing four bits strikes a balance between the write recovery time tWR and the tCL overhead. For example, with $n = 8$, the idea of splitting RESET phase is to have two half-RESET (hRESET) sub-phases. The first hRESET accounts for the "1" to"0"s in one half of the 8-bit group (b_0, b_2, b_4, b_8), while the second hRESET sub-phase accounts for the "1" to"0"s in the remaining half of the 8-bit group (b_1, b_3, b_5, b_7), as shown in Figure 8.

Figure 9 shows how the split RESET phase can reduce the write latency effectively. Again, taking "A512n8" as an example, the tWR decreases from 692ns to 200ns for the baseline1 without DSGB, and from 250ns to 138ns for the baseline2 with DSGB.

5.2.2. Compression-Enabled Dual-Write-Speed Mode In spite of the optimizations discussed so far, the write latency of ReRAM is still many times that of DRAM. We can reduce the number of hRESET phases if the data word has few "0"s. This can be easily achieved with selective data inversion (see example in Figure 10). Similar approaches were also proposed for PCM devices to reduce write energy [5] and to tolerate stuck-at faults [41]. We could do the same, but assuming a single inversion bit for n bits of data yields a storage overhead of 1/n since we need an additional flag bit to indicate whether the (n-1)-bit group is flipped or not, as shown in Figure 10 . Given that n is typically ≤ 16, the storage overhead will be at least 6.25%, which reduces the effective cost-per-bit accordingly. In order to avoid a fixed overhead of 6.25% for every cache line in memory, we propose a compression based encoding technique. A compressed line has spare room to accommodate inversion flag bits. Therefore, if the line is compressible, we perform inversion and reduce the number of hRESETs. Figure 11 is an overview of the compression/encoding flow in our

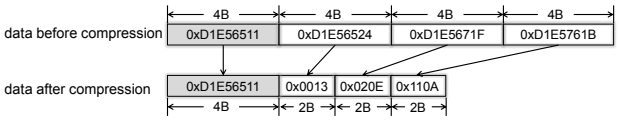

Figure 12: A delta compression example: B4Δ2 compression

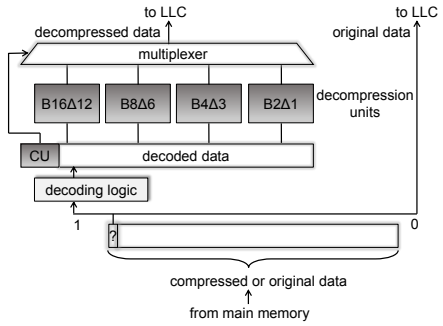

Figure 13: An overview of the decoding/decompression flow

Table 2: Power/Area overhead of hardware implementations

Metrics	Compression	Decompression	Encoders	Decoders
Area (μm^2)	9480	3460	2518	786
Power (mW)	4.41	2.43	4.77	3.58

framework. It works as follows:

- Every cache line is assigned a bit to represent if it is compressed or not. This lowers the storage overhead to under 0.2%. We assume BΔI as our compression algorithm [31]. BΔI is a low-latency algorithm that has sufficiently high compressibility for our purpose.
- If a line cannot be compressed, then it does not use data inversion to reduce the number of hRESET phases.
- If a line can be compressed, then at most n-1 bits of the compressed cache line are placed in every n-bit group. At least one bit is available per n-bit group to indicate if the data has been inverted or not.
- The compression bit per cache line can be stored in virtual memory and cached in the LLC, similar to other approaches [30, 43].
- On a write, the memory controller performs the compression to determine how data must be organized and how many hRESETs must be issued. The compression bit must be updated in the LLC or memory (off the critical path).
- On a read, the memory controller must read the compression bit from the LLC or memory in addition to reading the cache line from memory. The compression bit is required before the read data can be interpreted and sent to the processor. The access of the compression bit can be performed in parallel and therefore hidden. But such compression metadata accesses can increase memory traffic and hence introduce small slowdowns.

Note that we are not using compression to improve capacity or bandwidth. Compression is only being leveraged to create space to store flip bits. A compressed block, after encoding, will have the same size as the original block, as shown in Figure 11.

As an example, Figure 12 demonstrates how B4Δ2 (4-byte base and 2-byte deltas) compression works. Here we assume a 16-byte cache line to save space, and the first 4 bytes in the cache line are treated as the base. The compressed data are represented with a 4-byte base and three 2-byte deltas, and thus have 10 bytes in total after compression. We choose BΔI compression as our primary compression unit because it is a simple yet efficient hardware compression technique that features low decompression latency and high compression ratio. We adapt the compression in a different and much simpler way from prior work using this technique [31] as our goals are different. Instead of reducing the average compression ratio, ideally we would like to maximize the percentage of compressible cache lines as long as every compressed cache line has a better compression ratio than $\frac{n-1}{n} \times 100\%$. For any given base size, we only need to choose one large delta size to maximize the chance of compressing a cache line as long as the following inequality is obeyed,

$$B + (\frac{64}{B} - 1)\Delta \leq \frac{n-1}{n} \times 64 - 1 \quad (1)$$

For example, if n=8 and B=4, we can simply choose $\Delta = 3$ instead of exploring different delta values (1,2,3) that target minimum compression ratio [31].

Figure 13 is an overview of the decoding/decompression flow in our framework. The header bit of every cache line from the read queue in the memory controller is checked: "0" indicates it is the original data; "1" indicates it requires decoding and decompression. The decompression unit is chosen based on the "CU" bits, and the corresponding effective data width will be determined.

Hardware Overhead Hardware implementations of compression/decompression units, and encoders/decoders are verified in behavioral Verilog by creating a testbench and simulating using Mentor Graphics Modelsim [26]. We further investigate the overheads in terms of power, area, and critical path by synthesizing our Verilog code using Design Compiler [48]. We choose a 45nm technology library and scale the metrics to 22nm technology accordingly. The compression units take less than 100ps to compress the cache line and this latency is negligible compared to the overall memory access latency. The latency of decompression, encoding, and decoding are also very small. The area and power overheads of the hardware implementations are summarized in Table 2. The storage overhead of our design is less than 0.2% (one bit per cache line).

5.3. Latency Aware Write Scheduling

Once some cache lines are encoded based on their compressibility, write latency is no longer homogeneous across cache lines. The tWR of ReRAM in "A512n8" can vary from 200ns to 105ns for a conservative baseline1 and 138ns to 74ns for an aggressive baseline2. While writes are often on the non-critical path, long write latency can still hurt performance by blocking a bank and delaying subsequent reads to that bank. We propose latency aware write scheduling for ReRAM.

In a typical memory system employing bidirectional data bus, writes get queued in a write buffer at the memory controller. Once this queue gets filled beyond a specified higher

Algorithm 1: Write latency aware memory scheduling

```
if number of writes in WQ > WQHT then
    //start WQ drain
    nW = 0;
    maxWi = M for all banks;
    while number of writes in WQ > WQLT do
        maxWi = maxWi − num of read requests in bank i;
        if maxWi≤ 0 then
            continue; //stop issuing write to bank i
        else
            issue one write ReqW to bank i;
            nW = nW + 1;
            if ReqW is fast write then
                maxWi = maxWi − 1;
            else
                //a slow write takes more than 1 slot
                maxWi = maxWi − W;
            end
        end
    end
else
    issue reads from RQ;
end
```

threshold (WQHT), the controller turns around the bus and flushes the writes until the number of pending writes is less than a specified lower threshold (WQLT). With our proposed scheme, a compressed/encoded cache line has at most half "0"s in any n-bit group and can be written at an improved latency in an ReRAM-based main memory. We refer to a write request with a compressible cache line as a fast write. In contrast, the write request with an incompressible cache line is referred to as a slow write. To avoid slow writes from blocking subsequent reads, when flushing writes, we consider two additional parameters: first, the latency of writes, and second, the number of outstanding requests to the bank to which that write is being scheduled. We schedule writes such that the slow writes are written to banks with lowest number of outstanding reads.

For example, any time when write queue drain starts, we assume 32 writes will be issued. Once these writes have been issued, the memory controller switches back to issuing reads. In the baseline design, it is possible that bank *i* finishes its writes at cycle 1000 and bank *j* finishes its writes at cycle 1320. If bank *i* has 10 pending reads and bank *j* has 20 pending reads, then this was not the best way to issue the writes as more reads to bank *j* will get stalled. Hence, when selecting 32 writes from the write queue, we pick writes such that more work is steered to banks that have shorter read queues than others. A detailed scheduling policy is described in Algorithm 1.

6. Results and Discussion

Simulation Setup We use GEM5 [2] as our simulation platform with the integration of NVMain [32], which is a cycle-accurate memory simulator for both DRAM and non-volatile memories. Table 3 shows the detailed baseline configurations of the processor and main memory in our experiments. A

Processor	4 cores; 3GHz; Out-of-order; issue width=8; 192-entry reorder buffer; 32-entry load-store queue
L1 I&D-cache	Private; 16KB per core; 2-way;
L2 cache	Shared; 4MB total; 16-way; 128-entry MSHR; 64-byte block size
eDRAM L3 Cache	32MB, 16-way 100 cycle access, tag store in SRAM
Memory Configuration	16GB; 2 ranks/DIMM; 8 chips/rank; 8 banks/chip;

Table 3: System Configuration

Latency	Symbol	DRAM	PCM	ReRAM
Activate to read/write	tRCD	15	48	18
Column address strobe	tCL	15	15	15
Column write to data valid	tCWD	13	13	13
Four activate windows	tFAW	30	50	30
Write to read	tWTR	7.5	7.5	7.5
Write recovery	tWR	15	300	-

Table 4: Memory Timing Specifications (unit:ns)

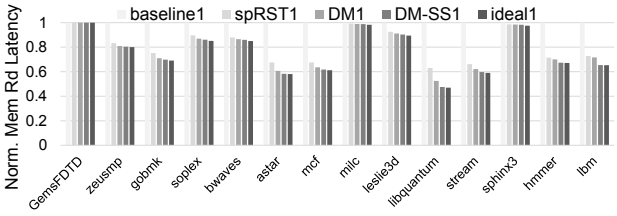

Figure 14: Normalized Average memory read latency of different optimizations applied to baseline1

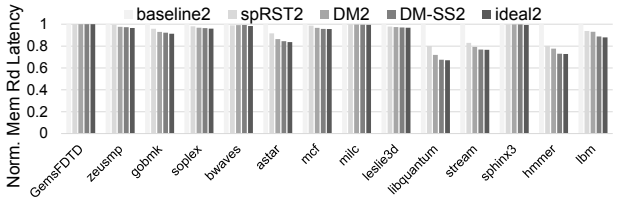

Figure 15: Normalized Average memory read latency of different optimizations applied to baseline2

DDR3-1066 SDRAM main memory is modeled based on the parameters from Micron's DDR3 technical specifications [28]. PCM is modeled similarly to Lee et al.'s work [19]. Table 4 lists the timing parameters (in nanoseconds) for DRAM, PCM, and ReRAM. The selected SPEC2006 CPU benchmark with reference input size [12] and STREAM with all functions [25] are evaluated as a multi-programmed testbench. We run all benchmarks for 500 million instructions for the cache warmup and then the following 100 million instructions for the statistics.

We evaluate our three optimization techniques on two baselines. The last digit "x" in the configurations indicates the baseline upon which these optimizations are applied to: "1" means they are applied to baseline1 without DSGB, and "2" means they are applied to baseline2 with DSGB.

- *baselinex*: The baseline ReRAM architecture with long write latency.
- *spRSTx*: Models the split RESET based on baselinex.
- *DMx*: Models dual-write-speed mode after applying compression and encoding techniques on spRSTx.
- *DM-SSx*: Models smart scheduling on DMx to make the

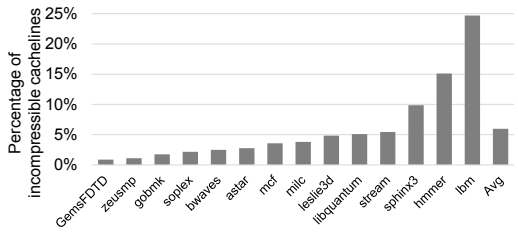

Figure 16: Percentage of incompressible cache lines across SPEC 2006 and STREAM benchmarks

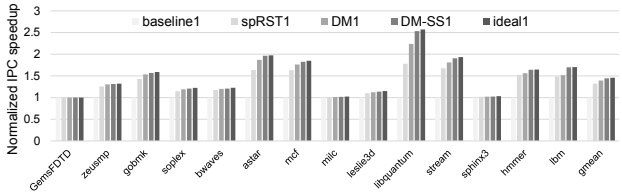

Figure 17: Normalized IPC speedup for SPEC 2006 and STREAM benchmarks applied to baseline1

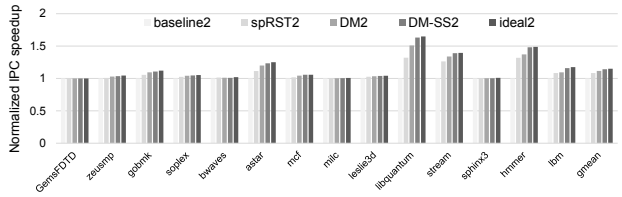

Figure 18: Normalized IPC speedup for SPEC 2006 and STREAM benchmarks applied to baseline2

memory controller aware of write latency.

- *idealx*: Models the oracle scenario assuming each write can be serviced at an improved latency that matches the faster write in DMx.

Memory Read Latency The effective memory read latency is one of the key performance metrics for a memory system and it is dominated by two components: (1) the queuing delay of a request in the memory controller; and (2) the time taken to serve the memory access to/from the bank. Our techniques improve both components.

Figure 14 shows the average memory read latency of these different configurations, with results normalized to the baselinex. The spRST1 alone reduces the average memory read latency by 27% relative to baseline1 since there is a more than 70% reduction of tWR. Figure 15 shows that the spRST2 is able to reduce the average memory read latency by about 7%, as the reduction of tWR is modest. Even based on the aggressive spRSTx design at a much improved write latency, our DMx and DM-SSx can further improve the read latency significantly, bringing it down close to that of the idealx. This works especially well for some write intensive workloads (e.g. *astar, libquantum, stream,* etc.). For example, the DM-SS1 reduces the average memory read latency by more than 50% for *libquantum* relative to baseline1.

To further examine the effectiveness of DMx, the percentage of incompressible cache lines across different benchmarks is plotted in Figure 16. Most applications have less than 5% incompressible cache lines, as our compression units are designed to optimize this metric. Some benchmarks, such as

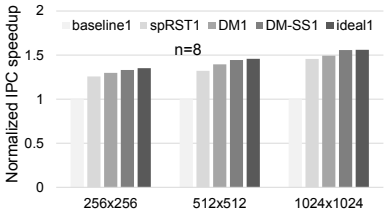

Figure 19: Performance sensitivity to mat size

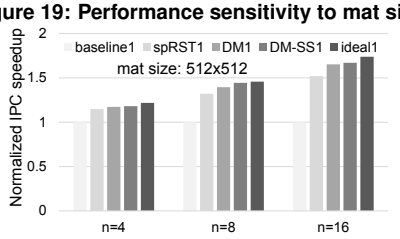

Figure 20: Performance sensitivity to *n*

hmmer and *lbm*, have higher than 10% incompressible cache lines, resulting in a significant gap from DMx to idealx, which in turn provides an opportunity for the DM-SSx model.

System Performance Figures 17 & 18 show the performance improvement of different memory configurations over the baselines thanks to the effect of reduced memory read latency. As expected, the spRST1 and spRST2 models improve performance by 32% and 8% over baseline1 and baseline2 respectively. The DM1 model further improves the IPC over spRST1 for some workloads with high write intensity (i.e., *gobmk, astar, mcf, libquantum, stream, hmmer, lbm*). The performance gap between DM1 and ideal1 for a few benchmarks (i.e., *libquantum, hmmer, lbm*, etc.) indicates that there is room to overcome the issues from a memory scheduling perspective. The DM-SS1 model is able to bring the performance within 1% of the ideal1 model.

Energy In the statistics of our exhaustive experiments, we observed that spRST1 and spRST2 can eliminate more than 30% of the hRESET sub-phases, and the average memory energy savings are estimated to be larger than 15%, given that the write energy of ReRAM contributes to more than half of the total memory energy consumption. The DM-SS1 and DM-SS2 models provide extra energy savings by limiting the number of cell flips. Therefore, the techniques we proposed are beneficial in terms of energy consumption of an ReRAM-based main memory.

Sensitivity to Mat Size Figure 19 shows the sensitivity of performance improvement to different mat sizes from 256×256 to 1024×1024. The trends for different mat sizes are similar. However, the improvement is higher for larger mat sizes. This is because a larger mat size leads to a worse baseline write latency and our optimization techniques can work more effectively on these memory systems. Specifically, the DM-SS1 model improves IPC by more than 50% over baseline1 on average.

Sensitivity to *n* Figure 19 shows the sensitivity of performance improvement to different values of n (the data fetch width from a mat) from 4, 8 to 16. Again, the trends for differ-

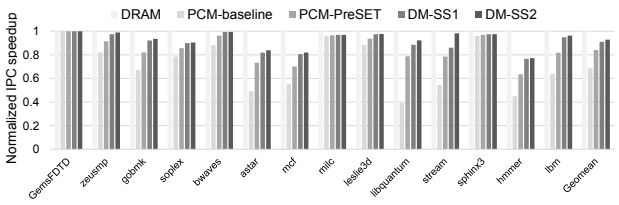

Figure 21: Performance comparison between different memory technologies

ent mat sizes are similar. It is observed that our optimizations work extremely well for a large n since a large n yields a higher baseline write latency. The average performance improvement of the DM-SS1 model over baseline1 are 21%, 45%, and 73% for n=4, 8 and 16.

Comparison with DRAM and PCM We compare our optimized designs DM-SS1 and DM-SS2 with a DDR3 DRAM design to see how close the design comes in terms of performance. We also model a baseline PCM design (PCM-baseline) and a state-of-the-art PCM design (PCM-PreSET) [33] for comparison. As seen in Figure 21, PCM-baseline impacts performance significantly, with a performance degradation of 32% compared to DRAM, which results from the high SET overhead of PCM. Although optimizations such as PCM-PreSET alleviate this problem, PCM-PreSET is still much slower than DRAM with an average performance degradation of 16%. Both of our optimized ReRAM designs out-perform the PCM-baseline and the PCM-PreSET. The simulation results show that average performance degradation for DM-SS1 and DM-SS2 are about 6% and 9% compared to DRAM, respectively. Thus, relative to DRAM, ReRAM is able to yield a significant benefit in terms of capacity, cost, and non-volatility, without a significant performance penalty.

7. Related Work

There have been many prior studies on architecting PCM as a main memory. Most of them deal with the shortcomings of phase change memory, such as long write latency, high write energy, and limited endurance. Several techniques were proposed to hide the long write latency of PCM. Qureshi et al. [34] proposed write pausing and write cancellation to minimize the chance of a read request getting blocked by a write request. Write truncation was used [17] to improve write performance in MLC PCM. PreSET was proposed [33] by exploiting the asymmetries in SET and RESET operations of PCM. PreSET issues the long-latency SET operations earlier than a write request actually arrives at the memory controller. However, in addition to its deleterious effect on endurance, it may generate more write traffic and saturate the low write bandwidth of PCM. Most of the previous approaches aim at hiding the long write latency while our design reduces the latency of most write requests. Moreover, our approaches have been designed specifically for ReRAM's crossbar architecture.

Writing a PCM cell requires large current, which limits the number of bits that can be written in parallel in a die. Hay et al. [11] improved write parallelism by tracking the number of modified bits and allocating power tokens smartly. Recently, Du et al. [8] proposed double XOR mapping (D-XOR) to distribute modified data bits among cell groups in a balanced way so that the overall write latency of serial SET/RESET operations is reduced. Motivated by the asymmetries in energy of RESET and SET operations, Yue et al. [55] proposed increasing parallelism within a bank by decreasing the number of *ones* in a write by flipping a word with more *ones*. Bit flipping was also studied to reduce write energy in PCM [5] and tolerate stuck-at faults [41]. In our work, we use bit-flipping along with compression to improve the RESET latency of writing a word in a crossbar.

Another emerging non-volatile memory technology being actively investigated is STT-RAM. Researchers have looked at techniques such as early write termination [58], hybrid SRAM/STT-RAM architecture [50] and read-preemptive write-buffer designs [46] to mitigate the long write latency of STT-RAM. Moreover, some prior work [44, 47] propose trading-off retention time to improve write latency. However, most of these studies target STT-RAM as cache replacement, and hence their architectures and specifications are very different from that of the main memory.

8. Conclusion

DRAM is facing many challenges beyond 16nm technology node. As changes to the core memory technology are very rare and considered a big leap in computing systems, it is critical to study all upcoming technologies and scrutinize every characteristic of them before embracing a technology. In this work, we explored the emerging ReRAM technology and its unique crossbar architecture. Although the crossbar is critical to achieving the best possible density, it poses serious challenges in terms of sneak current and voltage drop. To study their impact, we built a detailed modeling tool based on HSPICE and heavily modified CACTI. Based on our analysis using the tool, we showed that the data pattern in a crossbar architecture has a significant impact on the effective voltage across a cell, which in turn affects the overall write latency of ReRAM. To reduce sneak current, we proposed double-sided ground biasing and multi-phase write operations. The above two techniques together reduced the effective write latency from 692ns to 138ns, but the latency is still many times that of DRAM latency. To address this, we proposed and evaluated a compression based encoding scheme to reduce sneak current and improve voltage drop. The primary benefit of this approach comes from the encoding, with compression being used to include encoding bits without additional storage overhead. Together, this further reduces the write latency to 74ns based on the compressibility of a cacheline. Finally, we presented and evaluated a memory scheduling policy that considers the varying latency of ReRAM writes of compressed data along with activity of a bank when flushing writes to the memory. Overall, our architecture improves the performance of a system using ReRAM-based main memory by about 44% over a conservative baseline and 14% over an aggressive baseline, and has less than 10% performance degradation, compared to an ideal DRAM only system.

References

[1] R. Azevedo, J. D. Davis, K. Strauss, P. Gopalan, M. Manasse, and S. Yekhanin, "Zombie Memory: Extending Memory Lifetime by Reviving Dead Blocks," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2013, pp. 452–463.

[2] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The GEM5 Simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[3] G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. A. Lastras, A. Padilla, B. Rajendran, S. Raoux, and R. S. Shenoy, "Phase Change Memory Technology," *Journal of Vacuum Science and Technology B: Microelectronics and Nanometer Structures*, vol. 28, no. 2, pp. 223–262, 2010.

[4] G. Burr, K. Virwani, R. Shenoy, A. Padilla, M. BrightSky, E. Joseph, M. Lofaro, A. Kellock, R. King, K. Nguyen, A. Bowers, M. Jurich, C. Rettner, B. Jackson, D. Bethune, R. Shelby, T. Topuria, N. Arellano, P. Rice, B. Kurdi, and K. Gopalakrishnan, "Large-Scale (512kbit) Integration of Multilayer-Ready Access-Devices Based on Mixed-Ionic-Electronic-Conduction (MIEC) at 100% Yield," in *Symposium on VLSI Technology (VLSIT)*, June 2012, pp. 41 –42.

[5] S. Cho and H. Lee, "Flip-N-Write: a Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 347–357.

[6] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y.-J. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee, Y.-T. Lee, J. Yoo, and G. Jeong, "A 20nm 1.8V 8Gb PRAM with 40MB/s Program Bandwidth," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2012, pp. 46 –48.

[7] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee, "Better I/O Through Byte-Addressable, Persistent Memory," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 133–146.

[8] Y. Du, M. Zhou, B. R. Childers, D. Mossé, and R. Melhem, "Bit Mapping for Balanced PCM Cell Programming," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2013, pp. 428–439.

[9] R. Fackenthal, M. Kitagawa, W. Otsuka, K. Prall, D. Mills, K. Tsutsui, J. Javanifard, K. Tedrow, T. Tsushima, Y. Shibahara, and G. Hush, "A 16Gb ReRAM with 200MB/s Write and 1GB/s Read in 27nm Technology," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC),*, Feb 2014, pp. 338–339.

[10] B. Govoreanu, G. Kar, Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini, I. P. Radu, L. Goux, S. Clima, R. Degraeve, N. Jossart, O. Richard, T. Vandeweyer, K. Seo, P. Hendrickx, G. Pourtois, H. Bender, L. Altimime, D. Wouters, J. Kittl, and M. Jurczak, "10x10nm2 Hf/HfOx Cross-point Resistive RAM with Excellent Performance, Reliability and Low-Energy Operation," in *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*, 2011, pp. 31.6.1–31.6.4.

[11] A. Hay, K. Strauss, T. Sherwood, G. H. Loh, and D. Burger, "Preventing PCM Banks From Seizing Too Much Power," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2011, pp. 186–195.

[12] J. L. Henning, "Performance Counters and Development of SPEC CPU2006," *SIGARCH Comput. Archit. News*, vol. 35, no. 1, pp. 118–121, Mar. 2007.

[13] HP and SanDisk, "The Memristor Project." Available: http://www.businessweek.com/articles/2014-06-11/with-the-machine-hp-may-have-invented-a-new-kind-of-computer

[14] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda, "Dynamically Replicated Memory: Building Reliable Systems from Nanoscale Resistive Memories," in *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems*, 2010, pp. 3–14.

[15] ITRS, "International technology roadmap for semiconductors," 2013. Available: http://www.itrs.net/Links/2013ITRS/Summary2013.htm

[16] A. N. Jacobvitz, R. Calderbank, and D. J. Sorin, "Coset Coding to Extend the Lifetime of Memory," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 222–233.

[17] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. Childers, "Improving Write Operations in MLC Phase Change Memory," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, feb. 2012, pp. 1 –10.

[18] A. Kawahara, R. Azuma, Y. Ikeda, K. Kawai, Y. Katoh, K. Tanabe, T. Nakamura, Y. Sumimoto, N. Yamada, N. Nakai, S. Sakamoto, Y. Hayakawa, K. Tsuji, S. Yoneda, A. Himeno, K. Origasa, K. Shimakawa, T. Takagi, T. Mikawa, and K. Aono, "An 8Mb Multi-Layered Cross-Point ReRAM Macro with 443MB/s Write Throughput," in *IEEE InternationalSolid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2012, pp. 432 –434.

[19] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *Proceedings of the 36th annual international symposium on Computer architecture (ISCA)*. New York, NY, USA: ACM, 2009, pp. 2–13.

[20] H. Lee, Y. S. Chen, P. Chen, P. Gu, Y. Hsu, S. Wang, W. Liu, C. Tsai, S. Sheu, P.-C. Chiang, W. Lin, C. H. Lin, W.-S. Chen, F. Chen, C. Lien, and M. Tsai, "Evidence and Solution of Over-RESET problem for HfOX Based Resistive Memory with Sub-ns Switching Speed and High Endurance," in *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*, 2010, pp. 19.7.1–19.7.4.

[21] H. D. Lee, S. G. Kim, K. Cho, H. Hwang, H. Choi, J. Lee, S. H. Lee, H. J. Lee, J. Suh, S. Chung, Y. S. Kim, K. S. Kim, W. S. Nam, J. T. Cheong, J. T. Kim, S. Chae, E. Hwang, S. N. Park, Y. S. Sohn, C. G. Lee, H. S. Shin, K. Lee, K. Hong, H. G. Jeong, K. M. Rho, Y. K. Kim, S. Chung, J. Nickel, J. J. Yang, H. S. Cho, F. Perner, R. Williams, J. H. Lee, S. Park, and S. Hong, "Integration of 4F2 selector-less cross-point array 2Mb ReRAM based on transition metal oxides for high density memory applications," in *Symposium on VLSI Technology (VLSIT)*, june 2012, pp. 151 –152.

[22] M. Lee, C. B. Lee, D. Lee, S. R. Lee, M. Chang, J. H. Hur, Y. Kim, C. Kim, D. H. Seo, S. Seo, U. Chung, I. Yoo, and K. Kim, "A Fast, High-Endurance and Scalable Non-Volatile Memory Device Made from Asymmetric Ta2O5-x/TaO2-x Bilayer Structures," *Nature Materials*, vol. 10, pp. 625–630, Jul. 2011.

[23] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware Intelligent DRAM Refresh," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2012, pp. 1–12.

[24] T.-Y. Liu, T. Yan, Y. Chen, J. Lee, G. Balakrishnan, G. Yee, H. Zhang, J. Ya, A. Ouyang, S. T. *et al.*, "A 130.7mm2 2-Layer 32Gb ReRAM Memory Device in 24nm Technology," in *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC),*, Feb. 2013.

[25] J. D. McCalpin, "STREAM Benchmark." Available: http://www.cs.virginia.edu/stream

[26] Mentor Graphics, "Modelsim." Available: http://www.mentor.com/products/fpga/model

[27] Micron, "TN-41-01: Calculating Memory System Power for DDR3," http://www.micron.com/products/dram.

[28] Micron Corp., "Micron DDR3 SDRAM Data Sheet." Available: http://www.micron.com/products/dram/ddr3-sdram

[29] D. Niu, C. Xu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, "Design Trade-offs for High Density Cross-point Resistive Memory," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2012, pp. 209–214.

[30] G. Pekhimenko, T. C. Mowry, and O. Mutlu, "Linearly Compressed Pages: A Main Memory Compression Framework with Low Complexity and Low Latency," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2012, pp. 489–490.

[31] G. Pekhimenko, V. Seshadri, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Base-delta-immediate Compression: Practical Data Compression for On-chip Caches," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2012, pp. 377–388.

[32] M. Poremba and Y. Xie, "NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI),*, 2012, pp. 392–397.

[33] M. Qureshi, M. Franceschini, A. Jagmohan, and L. Lastras, "PreSET: Improving Performance of Phase Change Memories by Exploiting Asymmetry in Write Times," in *Proceedings of the International Symposium on omputer Architecture (ISCA)*, 2012, pp. 380–391.

[34] M. Qureshi, M. Franceschini, and L. Lastras-Montano, "Improving read performance of Phase Change Memories via Write Cancellation and Write Pausing," in *High Performance Computer Architecture (HPCA)*, 2010, pp. 1–11.

[35] M. K. Qureshi, "Pay-As-You-Go: Low-Overhead Hard-Error Correction for Phase Change Memories," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture(MICRO)*, 2011, pp. 318–328.

[36] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing Lifetime and Security of PCM-based Main Memory with Start-gap Wear Leveling," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 14–23.

[37] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable High Performance Main Memory System using Phase-Change Memory Technology," in *Proceedings of the 36th annual international symposium on Computer architecture (ISCA)*, 2009, pp. 24–33.

[38] P. Ranganathan, "From Microprocessors to Nanostores: Rethinking Data-Centric Systems," *Computer*, vol. 44, no. 1, pp. 39 –48, Jan. 2011.

[39] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ECP, not ECC, for Hard Failures in Resistive Memories," in *Proceedings of the international symposium on Computer architecture (ISCA)*, 2010, pp. 141–152.

[40] N. H. Seong, D. H. Woo, and H.-H. S. Lee, "Security Refresh: Prevent Malicious Wear-Out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping," in *Proceedings of the 37th annual international symposium on Computer architecture (ISCA)*, 2010, pp. 383–394.

[41] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "SAFER: Stuck-At-Fault Error Recovery for Memories," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2010, pp. 115–124.

[42] N. H. Seong, S. Yeo, and H.-H. S. Lee, "Tri-level-cell Phase Change Memory: Toward an Efficient and Reliable Memory System," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2013, pp. 440–451.

[43] A. Shafiee, M. Taassori, R. Balasubramonian, and A. Davis, "Memzip: Exploiting Unconventional Benefits from Memory Compression," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014.

[44] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan, "Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2011, pp. 50–61.

[45] J. Stuecheli, D. Kaseridis, H. C.Hunter, and L. K. John, "Elastic Refresh: Techniques to Mitigate Refresh Penalties in High Density Memory," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2010, pp. 375–384.

[46] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2009, pp. 239–249.

[47] Z. Sun, X. Bi, H. H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu, "Multi Retention Level STT-RAM Cache Designs with a Dynamic Refresh Scheme," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2011, pp. 329–338.

[48] Synposys, "Design compiler." Available: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler

[49] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2010, pp. 175–186.

[50] Z. Wang, D. A. Jimenez, C. Xu, G. Sun, and Y. Xie, "Adaptive Placement and Migration Policy for an STT-RAM-Based Hybrid Cache," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014.

[51] H.-S. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. Chen, and M.-J. Tsai, "Metal Oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[52] C. Xu, X. Dong, N. Jouppi, and Y. Xie, "Design Implications of Memristor-Based RRAM Cross-Point Structures," in *Design, Automation Test in Europe Conference Exhibition (DATE),*, 2011, pp. 1–6.

[53] D. H. Yoon, N. Muralimanohar, J. Chang, P. Ranganathan, N. P. Jouppi, and M. Erez, "FREE-p: Protecting Non-Volatile Memory Against Both Hard and Soft Errors," in *IEEE International Symposium onHigh Performance Computer Architecture (HPCA)*, 2011, pp. 466–477.

[54] S. Yu and H.-S. Wong, "A Phenomenological Model for the Reset Mechanism of Metal Oxide RRAM," *IEEE Electron Device Letters,*, vol. 31, no. 12, pp. 1455–1457, 2010.

[55] J. Yue and Y. Zhu, "Accelerating Write by Exploiting PCM Asymmetries," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 282–293.

[56] T. Zhang, M. Poremba, C. Xu, G. Sun, and Y. Xie, "CREAM: A Concurrent-Refresh-Aware DRAM Memory Architecture," in *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA),*, Feb 2014, pp. 368–379.

[57] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A Durable and Energy Efficient Main Memory using Phase Change Memory Technology," in *Proceedings of the 36th annual international symposium on Computer architecture (ISCA)*, 2009, pp. 14–23.

[58] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "Energy Reduction for STT-RAM Using Early Write Termination," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 2009, pp. 264–268.