Designing a Fast and Reliable Main Memory with Memristor Technology

Manjunath Shevgoor[†], Naveen Muralimanohar[‡], Rajeev Balasubramonian[†‡]
[†]University of Utah, [‡]HP Labs

Abstract

Several memory vendors are pursuing different kinds of memory cells that can offer high density, non-volatility, high performance, and high endurance. In this work, we focus on Memristor technology and identify some of the significant problems in state-of-the-art implementations. These problems include sneak currents during reads and non-uniformity in cell behavior within an array. These problems manifest as long read latencies, long write latencies, and high cache line error rates. To address these problems, we introduce multiple innovations to a memristor memory system: (i) We employ a background sneak current read that can be amortized across several other data reads from the same column, thus introducing "open-column" semantics for memristor array access. (ii) We also introduce a novel data mapping policy that reduces multi-bit error rates in cache lines. However, this policy also increases the average write latency for a cache line. (iii) We overcome this drawback by introducing data compression and avoiding poorly behaving cells during writes. The result is a memristor memory system that performs 12% better and has 30X lower probability of suffering a two-bit error compared to the baseline.

1. Introduction

Memory vendors are actively researching new ways to augment or even supplant DRAM technologies using novel technologies which provide higher memory capacities and non-volatility

Memristors have only been considered very recently. Memristors have a distinct density advantage because a single cell can approach an area of $4F^2$ and multiple cells can be vertically stacked with additional metal and oxide layers. They also have read latencies that are as low as 7.2 ns [2]. HP has announced a partnership with SK Hynix to build cross point memristor memories [12]. Some early architectural work [8, 9] carries out design space explorations to identify ideal memristor array layouts, and a recent paper [13] proposes a few optimizations to reduce read and write latencies for specific memristor design styles. This paper adds to this literature by defining some fundamental pieces of memristor microarchitecture that significantly improve its performance and reliability. The abstract summarizes the major contributions of a detailed paper that is currently under submission at ISCA.

2. Memristor/ReRAM Technology

Memristor, also referred to as ReRAM, is a stateful memory device built by sandwiching metal oxide material such as TiO2 or HfOx between electrodes [4, 11]. These devices have at least two stable states characterized by either low resistance or high resistance, and with the application of an external voltage, it can be switched from one state to another. Similar to PCM, a high resistance to low resistance transition is typically referred to as a SET operation and the reverse as a RESET operation.

In this work, we focus on a highly scalable, HfOx-based Memristor, which has an endurance of $> 10^{10}$ and can be switched at tens of nano seconds [4, 5]. By carefully architecting the memory array, HfOx-based Memristor can be used as a main memory along with or as a replacement to DRAM.

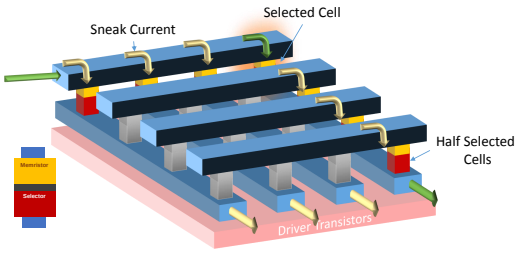

Figure 1: Crossbar array with each cell having a Memristor.

Crossbar Architecture: The current through a Memristor changes non-linearly with Voltage. This makes it possible to build a unique crossbar architecture. In a crossbar array (shown in Figure 1), a metal layer implements several wordlines and the next metal layer implements several bitlines. At every overlap point, the bitline and wordline are fused with metal-oxide material, forming a Memristor cell. If the voltage across a low resistance state cell exceeds a certain threshold, that cell essentially conducts current from the wordline to the bitline. Other cells that may not have sufficient voltage across them should ideally be non-conducting. In practice, a sneak current flows through such cells. By using a crossbar archtecture, it is possible to avoid the overhead of the access transistor associated with the "1T1R" design. In addition to the cell dimension, a Memristor array can also be scaled vertically by having multiple layers of cells. In a crossbar architecture, we can add a layer on top of an array by simply adding two metal layers with metal-oxide between them. Having four such layers can reduce the effective cell size to $1F^2$ [1, 7].

Memristor Reads/Writes: A Memristor crossbar allows reading a single cell at a time. If we are trying to read cell (i, j) in an array, a voltage V is applied to wordline i, and zero voltage is applied to bitline j. All other wordlines and bitlines are set to a voltage of V/2. As a result, assuming all cells apart from (i, j) are non-conducting, a voltage of V is applied across cell (i, j), making it a *fully selected cell* . It therefore conducts and a current is received at the bitline that corresponds to the resistance of the selected cell. Now the reason that all other cells are non-conducting is that they have a voltage of V/2 or 0 applied across them. Ideally, that voltage is low enough to keep the cell in non-conducting state. The cells that have 0 voltage applied to them are *unselected cells* and the cells that have V/2 voltage applied to them are *half-selected cells*. When writing cell (i, j), the same setup is used, except that a voltage V_W is applied across the fully-selected cell. To write the opposite value, V_W is applied to the selected bitline and 0 is applied to the selected wordline.

Sneak Current in a Crossbar: Unfortunately, cells have non-ideal behavior and conduct small amounts of current even if the voltage across them is less than the threshold voltage. This results in sneak currents through all the bitlines and wordlines. This results in IR-drops (voltage drops) along the wordlines, bitlines, and driver transistors, reducing the potential across the selected cell, and also increasing the current in the bitline of the selected cell.

Impact of Sneak Current: This leaked current in the bitline depends on the states of the half selected cells. For this reason, a memristor cell is read in two steps. First, the backgound current is read, where all rows and colums except the selected column are raised to V/2. This current is stored in a sense and hold circuit. The wordline voltage is then raised to V. The difference in the bitline current and the current in the sense and hold determines the value of the selected cell.

The IR Drop on the wordline because of the sneak currents decreases the voltage at the selected cell. The write latency of a 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 constants extracted from experimental results [6]. For example, a HfOx-based Memristor has demonstrated that a 0.4V reduction in RESET voltage may increase RESET latency by 10X [4].

3. Proposal

Crossbars are highly area efficient. However, they introduce a variety of problems stemming from sneak currents. These sneak currents impact energy, reliability (by introducing noise in reads and writes), and latency (by varying the voltage across cells and by requiring a multi-step read operation). In this paper, we propose architectural approaches to alleviate these concerns. We improve reliability, read latency, and write latency with three different techniques.

Reusing Background Currents: Reads are performed in two steps to overcome the effect of half selected cells on the read current. In the first step, the selected worline is biased with half select voltage and the selected bitline connects to the sample and hold circuit [3] . During the second step, the output from the sample and hold is fed to the sensing circuit to subtract the sneak current.

All prior work on memristors or ReRAM treat the array as a simple load store unit that performs all of these steps for every read/write operation. We make a key observation that even in the presence of variation in cells, the background current read for a column will closely resemble the background current read for other cells in the same column. Our detailed SPICE simulations show that the background current variation across the array is $< 6\mu A$ and within a column, the variation is less than $3\mu A$.

We propose to reuse the background current to read multiple cells from a single column. Based on SPICE simulations we found that the capacitor can retain its charge for upto 10μseconds or 32 reads, whichever comes first. This helps reduce latency, essentially halving memristor read latency every time the background current read is reused. For example, if a single read sense takes 50 ns, then the baseline has constant cache line read latencies of 100 ns, while open page accesses and non-open page accesses in our proposed model have read latencies of 50 ns and 100 ns, respectively. Architectural simulations using Simics show a performance increase of 13%.

Staggered Data Mapping for High Reliability: The data layout in memristors is such that each of the 512 bits in a cache line come from 512 different arrays. If all bits in a cache line occupy the same row and column in different arrays, such a mapping can lead to high uncorrectable error rates and alternative mappings are required. In a memristor crossbar, cells closer to the driver circuits are less vulnerable to noise from IR-drop than cells that are diametrically opposite. If we mapped data just as is done in any other memory, then cache lines which are mapped farther away from the drivers would have much higher error rates than cache lines which are closer. Such non-uniformity is not favorable for reliability because it concentrates errors to a few cache lines, rendering those cache lines vulnerable to uncorrectable multi-bit errors. Ideally, the probability of errors in every cache line should be uniform.

To achieve this, we introduce a data mapping policy that staggers the placement of every bit of a cache line in every array. For example if the first bit of a cache line is in the first column of the first array, the second bit is mapped to the second column of the second array, and so on. In essence, a cache line is made up of cells that have varying locations within crossbar arrays. With such a mapping, the probability of a single-bit or multi-bit error would be nearly uniform in every cache line. This staggered mapping reduces the probability of a two-bit error by 30x.

Improving Write Latency: As pointed out in Section 2, the voltage across a cell has a large impact on write latency. The write latency for each cell will therefore vary, depending on the position of the cell, the resistance states of neighboring cells, and the consequent IR-drops along the wordlines and bitlines. By staggering the data mapping, each cache line now has a bit in the farthest location, which also has the highest write latency.

We address this with an approach that relies on compression [10]. We augment the data mapping so that the first byte of a cache line is placed in the most favorable cells, and subsequent cache lines are progressively placed in less favorable cell locations. Before performing a write, a block is first compressed. As a result, a 64-byte block may now occupy (say) 16 bytes and the 128 most favorable cells available to it. It can therefore enjoy a lower write latency than an uncompressed cache line that necessarily uses the worst-case write latency. Thus, based on the compressed size of a block, the memory controller uses a range of latencies for its write operations. Compressing and remapping the data bits increases the performance by 15.9% and decreases the write energy by 49%, when compared to staggered mapping.

References

[1] Chevallier et al., "A 0.13um 64Mb Multi-layered Conductive Metal-oxide Memory," in *Proceeding of ISSCC*, 2010.

[2] S.-S. S. et al., "A 4Mb Embedded SLC Resistive-RAM Macro with 7.2ns Read-Write Random-Access Time and 160ns MLC-Access Capability," in *Proceedings of ISSCC*, 2011.

[3] Foltin et al., "Sensing Circuit for Resistive Memory," 2014, United States Patent, Number US): 700218780WO01.

[4] Govoreanu et al., "10x10nm^2 Hf/HfOx cross-point Resistive RAM with Excellent Performance, Reliability, and Low-energy Operation," in *Proceeding of IEDM*, 2011.

[5] D. Ielmini, S. Lavizzari, D. Sharma, and A. Lacaita, "Physical interpretation, modeling and impact on phase change memory (PCM) reliability of resistance drift due to chalcogenide structural relaxation," in *IEDM Technical Digest*, 2007.

[6] Lee et al., "Evidence and Solution of Over-RESET Problem for HfOx Based Resistive Memory with Sub-ns Switching Speed and High Endurance," in *Proceeding of IEDM*, 2010.

[7] Liu et al., "A 130.7mm2 2-Layer 32Gb ReRAM Memory Device in 24nm Technology," in *Proceeding of ISSCC*, 2013.

[8] D. Niu, C. Xu, N. Muralimanohar, N. Jouppi, and Y. Xie, "Design Trade-offs for High Density Cross-point Resistive Memory," in *Proceedings of ISLPED*, 2012.

[9] D. Niu, C. Xu, N. Muralimanohar, N. P. Jouppi, , and Y. Xie, "Design of Cross-point Metal-oxide ReRAM Emphasizing Reliability and Cost," in *Proceedings of ICCAD*, 2013.

[10] A. Shafiee, M. Taassori, R. Balasubramonian, and A. Davis, "MemZip: Exploiting Unconventional Benefits from Memory Compression," in *Proceedings of HPCA*, 2014.

[11] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. Williams, "The Missing Memristor Found," *Nature*, vol. 453, pp. 80–83, May 2008.

[12] X. Wang, Y. Chen, H. Li, D. Dimitrov, and H. Liu, "Bringing the memristor to market," 2010.

[13] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the Challenges of Cross-Point Resistive Memory Architectures," in *Proceedings of HPCA*, 2015.