# Improving Memristor Memory with Sneak Current Sharing

Manjunath Shevgoor
University of Utah, UT, USA
shevgoor@cs.utah.edu

Naveen Muralimanohar
HP Labs, CA, USA
naveen.muralimanohar@hp.com

Rajeev Balasubramonian
University of Utah, UT, USA
rajeev@cs.utah.edu

Yoocharn Jeon
HP Labs, CA, USA
yoocharn.jeon@hp.com

***Abstract*—**

**Several memory vendors are pursuing different kinds of memory cells that can offer high density, non-volatility, high performance, and high endurance. There are several on-going efforts to architect main memory systems with these new NVMs that can compete with traditional DRAM systems. Each NVM has peculiarities that require new microarchitectures and protocols for memory access. In this work, we focus on memristor technology and the sneak currents inherent in memristor crossbar arrays. A read in state-of-the-art designs requires two consecutive reads; the first measures background sneak currents that can be factored out of the current measurement in the second read. This paper introduces a mechanism to reuse the background sneak current measurement for subsequent reads from the same column, thus introducing "open-column" semantics for memristor array access. We also examine a number of data mapping policies that allow the system to balance parallelism and locality. We conclude that on average, it is better to prioritize locality; our best design yields a 20% improvement in read latency and a 26% memory power reduction, relative to the state-of-the-art memristor baseline.**

## I. Introduction

Memory vendors are actively researching new scalable non-volatile memory technologies that can augment or replace DRAM memories. Such emerging memories, e.g., PCM, STT-RAM, and memristors, have the potential to provide a big boost to servers handling big-data workloads by offering higher memory capacities and non-volatility. In addition, they also have the potential to dramatically simplify software overhead by enabling persistence at the main memory level [1].

However, many of these emerging NVMs also suffer from a variety of new problems that impact performance and energy-efficiency. While the use of NVMs as main memory is a very attractive prospect, the problems for each NVM have to be overcome. Most of the recent focus has been on PCM [2], [3], [4], [5] and STT-RAM [6], [7], [8]. For example, Lee et al. [2] architect PCM devices so they have energy and performance that are competitive with those of a DRAM system. Kultursay et al. [6] do the same for STT-RAM.

Memristors (or ReRAMs) 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 created in a vertical stack with additional metal and oxide layers, reducing the effective bit size to $< 4F^2$. They also have read latencies that can be as low as 7.2 ns [9] and have better on/off resistance ratio than STT-RAM. Initial industrial efforts are attempting to build memristor arrays that use crossbar architectures [10]. Some early architectural work [11], [12] 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 memristor read mechanisms that significantly improve its performance and energy-efficiency. We start with a tutorial on memristor cell and array design that helps identify key challenges and a reasonable baseline design. We then introduce new "open-column" semantics.

We observe that memristor array reads require a noisy read of current through the selected cell as well as a second read of background sneak currents to cancel out the noise. The latter measurement can be re-used when reading other cells in the same column of the array. Similar to DRAM's row buffer hit, this introduces the notion of a column hit in memristors. Managing the memristor column requires policies different from those of DRAM. We also observe that there can be a significant benefit from supporting parallel reads from many arrays; we therefore consider a number of address mapping policies. Our results show that performance and energy are best with a policy that places an entire subset of a page in a single array column.

## II. 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 [14], [15]. These devices have at least two stable states characterized by either low resistance or high resistance, which can be used to represent logical one and zero.

Resistive memories are broadly classified into either unipolar or bipolar devices based on their switching modes. In a unipolar device, change in state happens if an external voltage of specific magnitude and duration is applied across the device, whereas in bipolar devices switching of states also depends on the polarity of the external voltage [16], [17], [18], [9]. Even among bipolar devices, there are many metal oxide materials that exhibit resistance switching. In this work, we focus on a highly scalable, HfOx-based Memristor, which has an endurance of $> 10^{10}$ and the cell can be switched at tens of nano seconds [14], [19]. By carefully architecting the memory array, HfOx-based Memristor can be used as a main memory along with or as a replacement to DRAM.

## III. Array Architecture

Memristor arrays can either be designed as a grid with 1T1R cells [20] or as a dense *crossbar architecture* [21]. In

This work was supported in parts by the Department of Energy under Cooperative Agreement No. DE-SC0012199, NSF grants CNS-1302663 and CNS-1423583.

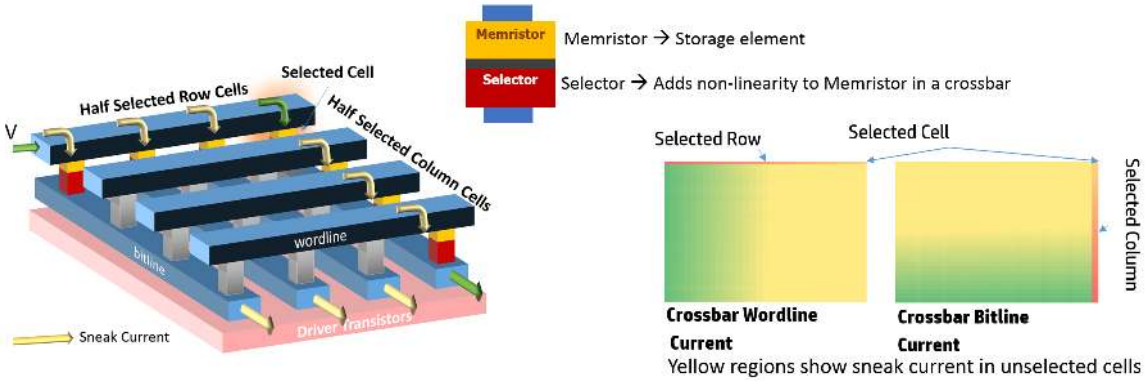


Fig. 1. Crossbar array with each cell having a Memristor storage element connected to a stateless selector device. The heat map shows the sneak current measured in the wordlines and bitlines in a crossbar array. The first row and the last column are selected to access the cell at the right most corner at the top.

a conventional design, each cell has a dedicated MOSFET transistor (a "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. This isolation provides better energy efficiency and access time compared to other array architectures. However, since resistive memories typically operate at a significantly higher current than DRAM, they require a large sized access transistor for each cell, making 1T1R cells less compelling for a cost conscious design. As we describe next, crossbar architectures have the potential to provide significantly higher densities.

### A. Crossbar Architecture

Most emerging resistive memory technologies change their state when a voltage equal to the switching voltage is applied across the cell. If the potential is less than the switching voltage, then the cell retains its state without getting disturbed. In contrast, in charge based technologies such as DRAM, any non-zero voltage across a cell can disturb its content. This property of resistive memory along with the non-linear nature of Memristor, where current changes non-linearly with voltage, make it possible to build a unique crossbar architecture.

In a crossbar array (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. Ideally, cells that do not have sufficient voltage across them should be non-conducting. In practice, a sneak current flows through such cells.

With such a design, we see that all cells are interconnected to form a dense grid without any access transistors. By eliminating access transistors, cells in a crossbar achieve the smallest theoretical size of $4F^2$. A cell size of $4F^2$ is literally the area under the cross-section of a minimum sized wordline and bitline. In addition, Memristor cells employ different fabrication steps from transistors; therefore, 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 Memristor-based main memory.

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 $1F^2$ [22], [23].

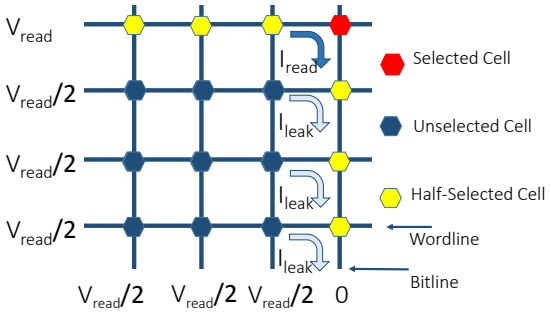


Fig. 2. A read operation to the top right corner cell in a 4×4 crossbar array.

### B. Memristor Reads/Writes

A Memristor crossbar allows reading a single cell at a time. We illustrate this in Figure 2 with an example. 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* (the red cell in Figure 2). 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 (the blue cells in Figure 2) are *unselected cells* and the cells that have V/2 voltage applied to them (the yellow cells) are *half-selected cells*.

With this ability to read a single cell at a time, over-fetch [24] in Memristor memories can be eliminated. When reading a cache line, 512 separate 1-bit reads can be performed, either in parallel or in series, either from one array or from 512 different arrays. Similarly, single-bit writes can also be performed. When writing cell $(i, j)$, the same setup as Figure 2 is used, except that a voltage $V_W$ is applied across the fully-selected cell (and voltage $V_W/2$ is applied to all other wordlines and bitlines). To write the opposite value, $V_W$ is applied to the selected bitline and 0 is applied to the selected wordline.

### C. 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. Further, this results in IR-drops (voltage drops) along the wordlines and bitlines. Therefore, the voltage applied across every cell is not the idealized V, V/2 or 0 volts.

Memristor 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 and unselected 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* ($\kappa$). The higher the $\kappa$, the lower the sneak current.

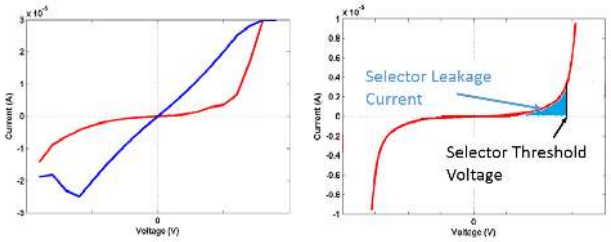


Fig. 3. Typical I-V curves of Memristor (left) and selector (right) elements in a cell. The blue state in the left corresponds to logic 1 and the red corresponds to logic 0.

Many recent Memristor prototypes employ a dedicated selector or bi-polar diode in each cell to improve $\kappa$, as shown in Figure 1 [16], [25], [23]. A selector is a state-less device, which can be laid on top of the Memristor material, without requiring additional area. In this work, we model a NbO based selector, which has a highly non-linear curve [26]. An ideal selector should act as a perfect switch with zero leakage current when the applied voltage across it is less than the selector threshold voltage. However, as shown in Figure 3, it is inevitable that some amount of leakage current flows through the selector, contributing to the sneak current in a crossbar.

### D. Impact of Sneak Current

Although a crossbar architecture is best suited for building dense memories, sneak current places strict constraints on read/write margins and array specifications. Most Memristor memories, even with a dedicated selector in each cell, have only finite non-linearity. Hence, sneak currents flowing through them pose a number of challenges, opening up new research possibilities for architects.

*1) Crossbar Size::* As discussed before, large crossbars and consequently large sneak currents can increase noise during reads and writes. In addition, the amount of sneak current ultimately determines the energy efficiency, access time, and area of an array. For example, sneak currents in a large array can cause a large IR-drop across half-selected cells. We have to therefore provide a higher voltage at the driver so that even after IR-drops, the fully-selected cell sees a sufficiently high voltage. Without this, the write may not succeed (*write failure* [11]). But a high voltage at the driver also results in high voltage at nearby half-selected cells. This can lead to *write disturbance* [11]. This inherent conflict requires an array to be moderately sized. For a given read or write bandwidth of a crossbar array, Figure 4 shows the increase in sneak current as the array size increases from 16x16 to 256x256 configurations. For a given array size, having the same number of rows and columns minimizes the half select path, and hence we only consider arrays with square aspect ratios. A detailed description of modeling and methodology for calculating the sneak current can be found in Section V.

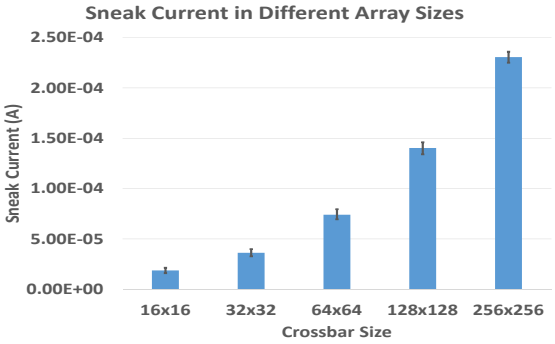


Fig. 4. Sneak current for different crossbar sizes.

*2) Array Bandwidth::* In a conventional memory array with access transistors per cell, activating a row results in reading out all cells in a row. Such arrays, e.g., with DRAM or PCM, have dedicated sense-amplifiers for every bitline or groups of bitlines. An array therefore has inherently high read bandwidth and row buffers are a natural extension to such arrays.

A memristor crossbar, on the other hand, does not lend itself easily to high-bandwidth access. As seen in the example in Figure 2, activating a single wordline and bitline in an $n \times n$ array allows us to read a single cell value, while introducing $2n-2$ half-selected cells. Alternatively, we could add another sense-amplifier circuit to the array and set another bitline to 0 volts. This would allow us to read two bits in a row, but would grow the number of half-selected cells to $3n-4$. This leads to two important drawbacks. First, the energy for the array read goes up dramatically because of the higher number of half-selected cells. This requires larger drivers, which in turn impacts density and cost [11]. Alternatively, the driver size can be kept constant, but $n$ can be reduced – this too reduces density. Second, the sense-and-hold circuits that constitute the sense-amplifier are large and doubling this number per array will also negatively impact density. In fact, we expect that future Memristor devices will likely share one sense-amplifier among 16-64 different arrays. Third, as discussed previously, higher sneak currents will exacerbate the write failure and write disturbance problems.

For these reasons, we expect that future cost-sensitive memristor devices will likely have very low bandwidth per array. In this work we investigate single-bit read/write per array although the proposed techniques can be extended to multi-bit accesses.

*3) Read Complexity::* For a given read voltage, a Memristor cell typically conducts $< 5\mu A$ in its off state ("0") and $> 15\mu A$ in its on state ("1"). While $\sim 3\times$ on/off ratio can offer an excellent read margin, when cells are connected in a

crossbar fashion with the afore mentioned biasing scheme, the selected bitline will carry a sneak current of $134 - 146\mu A$ (in a 128x128 crossbar) in addition to the cell current as shown in Figure 4. This variation is due to a number of factors such as the distribution of 1s and 0s across an array, selector threshold and leakage variation, and Memristor on/off resistance variation. As variance in sneak current is greater than the difference between on and off currents, a simple sense-amplifier with a single reference current cannot faithfully detect the state of the memory cell. Furthermore, since sneak current can vary based on the data stored in the array, it is critical to architect the sensing circuit such that we have enough noise margin to differentiate sneak current from the total read current.

Crossbar arrays typically require a complex two level sensing circuit in which, a read operation is split into three parts. First, similar to DRAM, bitlines are precharged before a read operation. Second, a half-select voltage is applied to the selected row to measure the background current in the present state of the array. In other words, a voltage of V/2 is applied to the selected row and a voltage of 0 is applied to the selected column. When this is done, the bitline current represents the current contributions of all the half-selected cells in that column. A special sample and hold circuit that is part of the sense amplifier measures this background current, and stores it in a capacitor. Finally, a normal read operation is performed, that factors out the sampled background current. This approach is therefore trying to eliminate the noise from the problem mentioned previously. There are two downsides to this approach: first, read latency is much larger than DRAM, which uses a simple sense-amplifier. Second, the silicon area overhead of the variation aware circuits is significantly higher, limiting the number of sense-amplifiers per array. As mentioned earlier, a single sense-amplifier will likely be shared among many arrays. In comparison, DRAM has a sense-amplifier for every bitline in the array. Hence, techniques that reduce sensing overhead in memristors are critical to improve read latency and memory bandwidth.

## IV. Proposal

In the previous section, we have shown that the non-linearity of a memristor cell makes it a good fit for a crossbar architecture. Crossbars are highly superior in terms of density, which is why they are attractive for cost-sensitive memory products. However, crossbars introduce a variety of problems stemming from sneak currents. These sneak currents impact energy and latency (by varying the voltage across cells and by requiring a multi-step read operation). In this paper, we examine architectural approaches to alleviate these concerns and improve read latency and energy.

Our baseline is a memristor memory system that, like DRAM, is composed of channels, DIMMs, ranks, and banks. We assume that a bank is composed of several crossbar arrays. On a 64-byte cache line request, single-bit reads are performed in parallel on 512 different arrays. These 512 different arrays are scattered across the memristor chips that form the rank. Writes follow a similar process.

### A. Reusing Background Currents

Figure 5 shows the high-level block diagram of two-level sensing with a sample and hold circuit for crossbar

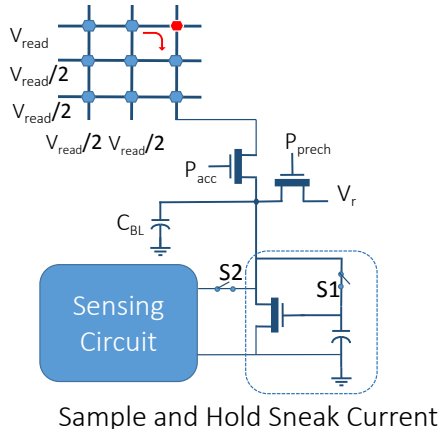


Fig. 5. Two level sensing with sample and hold circuit to cancel the effect of sneak current in reads.

memory [27]. During the first phase of a read operation, the selected wordline is biased with half-select voltage and the selected bitline connects to the sample and hold circuit with switch S1 closed and S2 open. The transistor, connected as a diode, charges the holding capacitor based on the potential at its gate and source, which in turn is a function of the leakage current flowing through the bitline. During the second step, S1 is opened to isolate the holding capacitor from the selected column and S2 is closed to initiate sensing. Since the transistor gate potential is maintained by the capacitor, the same current equivalent to the sneak current flows through the drain. When the selected wordline voltage is increased to full read voltage, the selected bitline current goes up. However, the effective current fed to the sense-amp will be the difference between the total bitline current in the second phase and the drain current induced by the capacitor.

In a crossbar array, biasing unselected rows and columns to half select voltage can help avoid read or write disturbance and reduce sneak current. To half-bias all unselected rows and columns, we need a voltage source multiplexer for every row and column, which can increase the area of the crossbar. Hence, it is better to limit the half-biased rows and columns as much as possible to improve density.

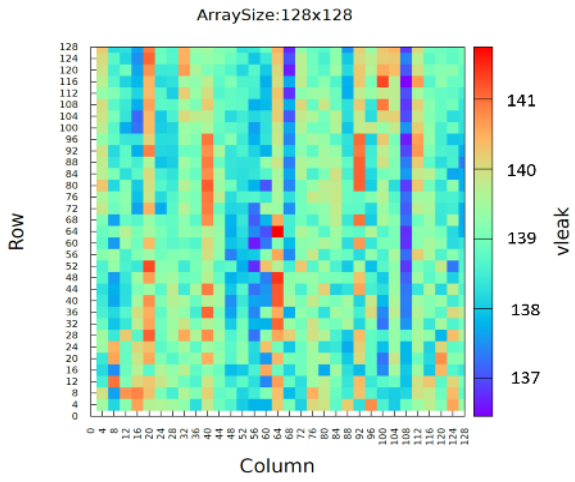


Fig. 6. Background current measured for cells at various location in a crossbar array. The background current varies from 136 to 142 $\mu A$.

Most prior work on memristors treat the array as a simple

load store unit that performs both sampling and sensing for every read operation [1]. 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.

Figure 6 shows a heat map of background currents for different cells in a crossbar. The figure is based on the assumption that 1s and 0s are randomly distributed with equal probability. The selector, memristor, and cell variation parameters used to model this are discussed in Section V. Overall, the background current variation across the array is $< 6\mu A$ and within a column, the variation is less than $3\mu A$. When sensing current along the bitline, the half selected row cells mainly contribute to the reduction in voltage across the selected cell, whereas the half selected column cells contribute to the total sneak current in the selected column. Hence, even if we do not assume random distribution of 1s and 0s, the variation within a column is always smaller as cells that are contributing to the sneak current are mostly the same. The main factor that affects the variation of sneak current within a column is the current coming from the floating wordlines and bitlines. As we half-bias more unselected rows and columns, the variation of background current within a column goes down further.

Based on this observation, we propose to reuse the background current to read multiple cells from a single column. As in the baseline, a cache line request will first involve a read of the background current for that column (in 512 different arrays). This is followed by a read of the fully-selected cell corresponding to the cache line being read. Once this is done, the sample and hold circuit continues to retain its stored charge. Based on SPICE simulations we found that the capacitor can retain its charge for upto $10\mu$ seconds or 32 reads, whichever comes first.

Hence, if subsequent requests are made to other cells in the same column, the background current read can be elided. The capacitor in the sample and hold circuit continues to assist the next many reads from that same column, just as a DRAM row buffer continues to assist subsequent reads from the same row. 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 in our proposed model, the read latency can be either 100 ns or 50 ns, depending on whether a background current read is required or not.

This reuse of the background current re-introduces the concept of an *open page access*, which was deemed unnecessary for memristors. Physically, this is very different from DRAM open page access – instead of storing a row's worth of data in a "cache" with fast access, the memristor is simply storing background current in a single capacitor. While many of the semantics for open page access are similar to those for DRAM, there are also significant differences:

1) There is an expiration time for the capacitor in the sample and hold circuit because its charge gradually leaks away.
2) The number of reads and writes that can be serviced by the capacitor is limited.
3) A write to any cell in that column renders the background current measurement invalid. Based on the process variation in the cell that got written, its impact on the bitline sneak current can vary. In other words, open page access only works for reads.
4) The number of "open" columns in the memristor memory system can be much larger than the number of open rows in a DRAM system.
5) The organization of data in arrays must be transposed so that consecutive cache lines are made to share the same column and not the same row (as in DRAM). This helps exploit spatial locality for open page accesses.

The open page access described so far offers latency and energy savings. However, if workloads exhibit low levels of spatial locality, then both benefits are small. Instead, to save energy in a guaranteed manner, we can configure open page access to benefit each individual cache line read. When reading a cache line, we can fetch (say) 2 bits of that cache line from the same column of the same array. Since an activated column can read only one bit at a time, the 2 bits from a given column will have to be read sequentially. But this process is performed across 256 different arrays in parallel. The result is a higher latency of 150 ns for the cache line read, but a reduction in energy dissipation from sneak currents. If a subsequent cache line access is to the same column, it has an access latency of 100 ns. However, increasing the memory latency in this manner results in longer execution times and more energy dissipated in the rest of the system. We therefore do not consider such models further in this paper.

If an application does exhibit spatial locality, we will see multiple back-to-back accesses to the same array and column. Even though our open-page mode avoids multiple background current reads, throughput is limited because the second array read cannot begin until the first array read has completed. This delay is much higher than the delay seen in DRAM for open-page accesses. In DRAM, the second access does not perform an array read; it simply moves data from the row buffer to the output pins; this operation completes in about 5 ns. In memristors, two back-to-back accesses to the same array are separated by about 50 ns. Therefore, it is not evident that consecutive cache lines should be placed in the same column. If an application exhibits spatial locality, it might benefit by placing consecutive cache lines in different arrays so that both cache lines can be fetched in parallel. In short, the address mapping policy can possibly have a significant impact on memristor throughput. We investigate this further in Section VI.

## V. Methodology

In Memristor memory, both access latency and energy are dominated by the cell and crossbar array, whereas in charge based technologies such as DRAM, the overhead of accessing a *mat*[2] itself is small compared to the routing and IO overhead. To evaluate the impact of the proposed techniques on Memristor performance and power, it is critical to have an accurate model of a crossbar array. For this work, we

[1]Note that sample and hold circuit is also used for writes to ensure current compliance. For this work, we focus on improving read latency and energy.

[2]Mat is the basic building block of a DRAM bank.

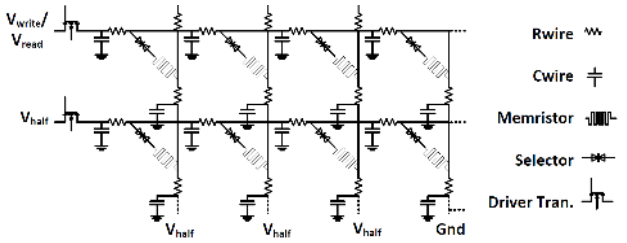


Fig. 7. Crossbar components modeled.

built a tool that takes array size, cell parameters, and process technology as input to generate a complete HSPICE netlist, which is then used to calculate output wordline current, sneak current, voltage drop across the row, and transient delay.

Figure 7 shows the components modeled to simulate a crossbar array. We use HfOx-based cell parameters from Lee et al. [28] and selector parameters from Pickett et al. [26]. The memristor element is modeled as a voltage controlled current source and the selector as a current controlled voltage source. For transistor parameters in peripheral circuits, we use AMI $250nm$ technology. For all our analysis, we consider only a single layer crossbar array and access only one bit per array. A detailed list of selector, memristor, and wire parameters is tabulated in Table I. The sneak current in Figure 4 and heatmap in Figure 6 are based on HSPICE simulations using these parameters.

For performance analysis, we run our simulations for a total of 250M instructions. We use Simics [29] functional simulator and augment its trans-staller module with a detailed memristor timing model based on USIMM [30], including memory controller overhead and its queuing delay. The CPU we simulated consists of eight out-of-order cores with 32MB of shared LLC, similar to [31]. The memristor memory system has two DDR channels with each channel having two ranks. We use workloads from SPEC2006 benchmark suite. Simulator parameters are summarized in Table II.

We experimentally determine the best address mapping policy for our memristor baseline. We map successive cache lines to different channels, ranks, banks and sub-banks. The sub-bank is XORed with the column in order to further reduce sub-bank conflicts [32]. Each time a cache line needs to be read, the background current is first read, followed by the actual read. This baseline optimizes parallelism when servicing memristor reads.

The sample and hold circuit that provides the background current is shared between all the crossbars in a subarray. In our design we assume 8 Banks per rank, 32 sub-banks, and 64 subarrays per sub-bank. Hence we have a total of 16K sense and hold circuits per rank. A subarray here is defined as a set of crossbars that share a two-level sensing circuit. A sub-bank consists of a set of subarrays from which a cacheline is retrieved.

## VI. Results

Figure 8 shows the performance increase when a baseline memristor system is compared with a DRAM system, and with the proposed system that re-uses the background current.

The first bar (DRAM) shows the performance increase when a DRAM based memory system is used. The address mapping policy used here maps an entire OS page to a single row, thus maximizing row buffer hits. The DRAM system is intended to show the potential room for improvement in an idealized memory system with latencies as low as that of DRAM. Of course, a DRAM only system will offer much lower memory capacity and will be a poor fit for big data workloads because of large page fault rates. This study does not consider the effect of memory capacity on page fault rates, so the first bar in Figure 8 is an optimistic upper bound.

The second bar (32ReUse) shows the performance increase when the proposed memristor system is used. As discussed in the proposal section, we change the default address mapping policy such that successive cachelines get mapped to rows in the same column of a memristor crossbar, i.e., an entire OS page maps to a single column in a sub-bank. This policy enables us to exploit the spatial locality of workloads to reuse background current measurements. We re-use the background current for 32 successive reads to the same column of the same sub-bank, as long as they occur within 10 $\mu s$.

Zeusmp sees the largest performance improvement because of a combination of high column hit rates and long gaps between successive accesses to the same page. Compared to the baseline memristor system, reusing the background current increases performance by 8.3%, on average across all benchmarks. The DRAM system is 34.8% better than the baseline system, however it is only 24.4% better than the proposed memristor system.

Memristor latency is a combination of the time it takes to sense the background current, and the time it takes to sense the data stored in the cell. Figure 9 shows the memory latencies for the baseline system (NoReUse), the proposed system (32ReUse), and a DRAM system. Memory latency is

TABLE I. Parameters in the crossbar Array Model

| Metric | Description | Value or Range |
|---|---|---|
| $A$ | Crossbar size: $A$ wordlines $\times A$ bitlines | $128 \times 128$ |
| $n$ | Number of bits to read/write | 1 |
| $I_{on}$ | Cell current of a LRS Memristor | $15uA$ |
| $I_{off}$ | Cell current of a HRS Memristor | $4uA$ |
| $R_{wire}$ | Wire resistance between adjacent cells | $8\Omega$ |
| $V_{th}$ | Selector threshold voltage | $1.5V$ |
| $\sigma_{vth}$ | Selector voltage variation | $15\%$ |
| $Ileak$ | Selector half-select leakage | $1\mu A$ |
| $\sigma_{leak}$ | Selector half-select leakage variation | $.1\mu A$ |
| $K_r$ | Selector $i_{leak}$ variation | $15 - 30\%$ |
| $V_W$ | Full selected voltage during write | $3.6V$ |
| $V_R$ | Read voltage | $2.0V$ |

| Processor | |
|---|---|
| Core Parameters: | UltraSPARC III ISA, 8-core, 3.2 GHz, 64-entry ROB, 4-wide OOO. |
| **Cache Hierarchy** | |
| L1 I-cache | 32KB/2-way, private, 1-cycle |
| L1 D-cache | 32KB/2-way, private, 1-cycle |
| L2 Cache | 8MB 64B,8-way, shared, 10-cycle |
| Coherence Protocol | Snooping MESI |
| **Memory** | |
| Memristor Frequency | 1600 Mbps |
| Channels, ranks | 2 channels, 2 ranks/channel, |
| Banks, Sub Banks | 8 banks/rank 32 sub banks/bank |
| Sub Arrays, Crossbars | 64/bank , 64 Crossbars/ sub array |
| Read Queue Length | 64 per channel |
| Memristor Timing | $BGCurrentSense$ = 50ns ns<br>$1 - bit/CrossbarRead$ = 50 ns<br>$2 - bit/CrossbarRead$ = 100 ns |

TABLE II. Simulator and Memristor Timing parameters.

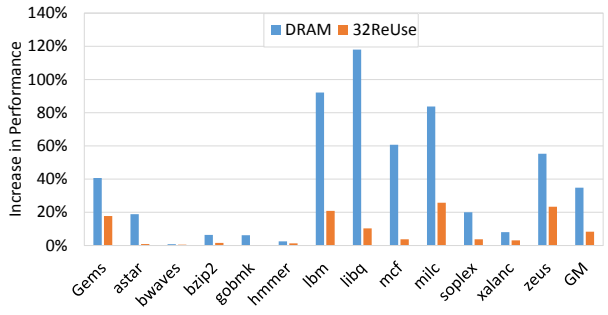


Fig. 8. Performance impact of Background Current re-use

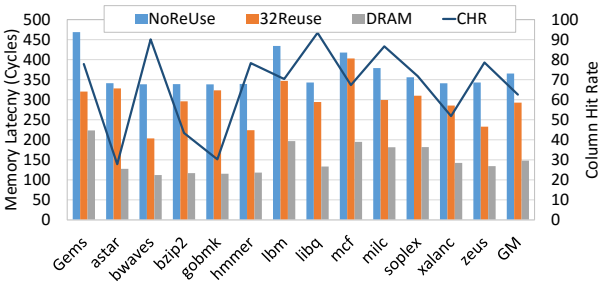


Fig. 9. Impact of BG current reuse on memristor read latency.

a combination of the memory core latency and the queuing delay in the memory controller. By re-using the background current, we are able to reduce total memristor memory latency by 20%. The line titled CHR shows the column hit rates as seen when background current is re-used. The large number of sub-banks present in the memristor system reduces column conflicts, and hence is able to provide an average column hit rate of 67%. Applications that see high column hit rates, like Gems and bwaves, show the largest drops in memory latency, while astar and gobmk show the lowest.

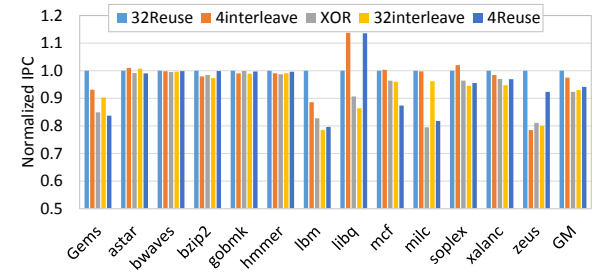


Fig. 10. Trading off locality for parallelism.

A key difference between a hit to a DRAM row-buffer and to a memristor crossbar column is the large latency difference, stemming from the absence of overfetch [24] in a memristor memory. Because DRAM senses more than a single cacheline, the CAS-to-CAS latency for DRAM is 5 ns. Memristors sense only a single cache line per read, and hence need to sense the resistance of the cell anew, even when the background current is re-used. For this reason, the CAS-to-CAS latency is much higher than DRAM (50ns in our evaluation). Workloads that have high locality as well as high Memory Level Parallelism (MLP) end up having multiple accesses to the same page at the same time. The high CAS-to-CAS latency of memristor memory can adversely impact the performance of such workloads even in the presence of high column hit rates. To address this, we investigate the trade-off between locality and MLP by evaluating different address mapping policies.

Figure 10 shows the performance of five different address mapping policies. 32ReUse maps the entire page to a single column in a sub-bank. 4interleave maps every fourth cacheline to the same sub-bank. XOR maps successive cache lines to different Channels, Ranks, Banks and Sub-banks. The sub-bank is XORed with the column in order to further reduce sub-bank conflicts [32]. 32interleave maps every 32nd cacheline in a page to the same sub-bank. 4ReUse maps 4 successive cachelines to the same bank, subsequent chunks of four cachelines are mapped to different Channels, Ranks and Banks in that order. 32ReUse, which maximizes locality, represents one end of the spectrum, while XOR, which maximizes parallelism, represents the other end of the spectrum. 4/32interleave and 4ReUse represent points in between. For all the workloads except libquantum, soplex and astar, 32ReUse has the highest performance. This is contrary to our observation in the memristor baseline, where the XOR scheme that maximizes parallelism does best. Therefore, our ability to reuse background currents has yielded a different optimal address mapping policy. On average, 32ReUse outperforms 4interleave by 3%, XOR by 8%, 32interleave by 7%, and 4ReUse by 6%.

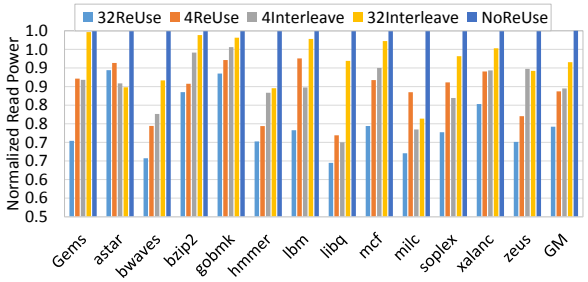


Fig. 11. Reducing average power by reusing background current.

Using the memristor parameters detailed in Table I and detailed HSPICE simulation, we determine that for a 128x128 array, reading the background current consumes 335 $\mu W$, while a full read takes 546 $\mu W$. This difference is due to the different row voltages required for background and full reads. Figure 11 shows the power consumption for the workloads we evaluated, normalized to the memristor baseline. On average, re-using the background current reduces Memristor read power by 25.8%. Other address mapping schemes that trade off locality for parallelism attain lower power savings. 4ReUse reduces average read power by 16.3%, 4Interleave reduces it by 15.5%, 32Interleave decreases it by 8.4%.

## VII. Related Work

A large body of work exists on leveraging emerging non-volatile memory to augment or replace DRAM main memory. Most of them focus on phase change memory, addressing its limitations such as long write latency, limited endurance, and high write energy [33], [5], [3], [4].

To hide the long write latencies of PCM, Qureshi et al. [3] proposed write pausing and write cancellation. Jiang et al. [5] proposed write truncation to improve the write performance in MLC PCM. Cho et al. [33], flipped the cache line if it reduces the number of bits that needs to be written. This improves the endurance of PCM as well as write energy. Most of these approaches are orthogonal to the proposed techniques and are applicable to resistive memories such as memristors.

There are a number of device and circuit papers on ReRAM that discuss cell specification and circuit design to build

memristor memories [34], [11], [35]. The most recent work on ReRAM architecture is by Cong et al. [13], in which the authors discuss challenges in building crossbar memories focusing on multibit reads and writes within an array. They propose inverting a set of bits written to a crossbar to reduce the number of low resistance states within a crossbar and improve write performance. As discussed earlier, we target one bit operation per crossbar array due to its low voltage requirement and smaller driver size.

## VIII. Conclusion

Memristor is a promising emerging technology and a crossbar architecture is the best way to build dense memristor memory. In this work, we discussed key problems in designing a crossbar and proposed solutions to reduce read overhead. We enhance the two level sensing scheme typically employed for a crossbar, such that we reuse the background current read in the first step for subsequent reads. This reduces the effective read latency by 20% and memristor power by 25.8%. While the proposed scheme is beneficial for a majority of workloads, some benchmarks prefer more parallelism within the memory to improve performance. We investigated several address mapping schemes that exploit reusing background current for a different number of cachelines per page with varying levels of parallelism. We find that placing consecutive cache lines in the same column of the same array yields the highest performance and energy efficiency.

## References


[1] J. Condit *et al.*, "Better I/O Through Byte-Addressable, Persistent Memory," in *Proceedings of SOSP*, 2009.

[2] B. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *Proceedings of ISCA*, 2009.

[3] M. Qureshi, M. Franceschini, and L. Lastras, "Improving Read Performance of Phase Change Memories via Write Cancellation and Write Pausing," in *Proceedings of HPCA*, 2010.

[4] M. K. Qureshi, M. Franceschini, L. Lastras, and A. Jagmohan, "PreSET: Improving Read Write Performance of Phase Change Memories by Exploiting Asymmetry in Write Times," in *Proceedings of ISCA*, 2012.

[5] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in MLC phase change memory," in *Proceedings of HPCA*, 2012.

[6] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," in *Proceedings of ISPASS*, 2013.

[7] Q. Guo, X. Guo, R. Patel, E. Ipek, and E. Friedman, "AC-DIMM: Associative Computing with STT-MRAM," in *Proceedings of ISCA*, 2013.

[8] W. Xu, Y. Chen, X. Wang, and T. Zhang, "Improving STT MRAM storage Density through Smaller-Than-Worst-Case Transistor Sizing," in *Proceedings of DAC*, 2009.

[9] Sheu 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.

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

[11] 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.

[12] 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.

[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.

[14] Govoreanu et al., "10x10$nm^2$ Hf/HfOx cross-point Resistive RAM with Excellent Performance, Reliability, and Low-energy Operation," in *Proceeding of IEDM*, 2011.

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

[16] Kawahara et al., "An 8Mb Multi-Layered Cross-Point ReRAM Macro with 443MB/s Write Throughput," in *Proceedings of ISSCC*, 2012.

[17] Kim et al., "Low Power Operating Bipolar TMO ReRAM for Sub 10nm Era," in *Proceeding of IEDM*, 2010.

[18] Otsuka et al., "A 4Mb Conductive-Bridge Resistive Memory with 2.3GB/s Read-Throughput and 216MB/s Program-Throughput," in *Proceeding of ISSCC*, 2011.

[19] 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.

[20] Wong et al., "Metal-Oxide RRAM," in *Proceedings of IEEE*, 2012.

[21] M. M. Ziegler and M. R. Stan, "CMOS/Nano Co-Design for Crossbar-Based Molecular Electronic Systems," in *Proceeding of Transactions on Nanotechnology*, 2003.

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

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

[24] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. Jouppi, "Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores," in *Proceedings of ISCA*, 2010.

[25] Lee et al., "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*, 2012.

[26] M. D. Pickett and R. S. Williams, "Sub-100 fJ and Sub-nanosecond Thermally Driven Threshold Seitching in Niobium Oxide Crosspoint Nanodevices," in *Symposium on VLSI Technology*, 2012.

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

[28] 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.

[29] "Wind River Simics Full System Simulator," 2007, http://www.windriver.com/products/simics/.

[30] N. Chatterjee, R. Balasubramonian, M. Shevgoor, S. Pugsley, A. Udipi, A. Shafiee, K. Sudan, M. Awasthi, and Z. Chishti, "USIMM: the Utah SImulated Memory Module," University of Utah, Tech. Rep., 2012, UUCS-12-002.

[31] Cong et al., "Overcoming the Challenges of Crossbar Resistive Memory Architectures," in *Proceedings of HPCA*, 2015.

[32] Z. Zhang, Z. Zhu, and X. Zhand, "A Permutation-Based Page Interleaving Scheme to Reduce Row-Buffer Conflicts and Exploit Data Locality," in *Proceedings of MICRO*, 2000.

[33] S. Cho and H. Lee, "Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy, and Endurance," in *Proceedings of MICRO*, 2009.

[34] J. Liang and H.-S. P. Wong, "Cross-Point Memory Array Without Cell Selectors- Device Characteristics and Data Storage Pattern Dependencies," *IEEE Transactions on Electron Devices*, vol. 57, 2010.

[35] C. Xu, X. Dong, N. P. Jouppi, , and Y. Xie, "Design Implications of Memristor-Based RRAM Cross-Point Structures," in *Proceedings of DATE*, 2011.