A Low-Power Hybrid Reconfigurable Architecture For Resistive Random-Access Memories

Miguel Angel Lastras-Montaño Amirali Ghofrani Kwang-Ting Cheng
Electrical and Computer Engineering Department
University of California, Santa Barbara
{mlastras, ghofrani, timcheng}@ece.ucsb.edu

ABSTRACT

Access-transistor-free memristive crossbars have shown to be excellent candidates for next generation non-volatile memories. While the elimination of the transistor per memory element enables higher memory densities, it also introduces parasitic currents during the normal operation of the memory that increases both the overall power consumption of the crossbar, and the current requirements of the line drivers.

In this work we present a hybrid reconfigurable memory architecture that takes advantage of the fact that a complementary resistive switch (CRS) can behave both as a memristor and as a CRS. By dynamically keeping frequently accessed regions of the memory in the memristive mode and others in the CRS mode, our hybrid memory offer all the benefits that a memristor and a CRS offer individually, without any of their drawbacks.

We validate our architecture using the SPEC CPU2006 benchmark and found that our hybrid memory offers average energy savings of 3.6x with respect to a memristive-only memory. In addition, we can offer a memory lifetime that is, on average, 6.4x longer than that of a CRS-only memory.

1. INTRODUCTION

Purely memristive or access-transistor-free (ATF) non-volatile crossbar arrays have been recognized as potential alternatives to existing non-volatile and volatile memories: They can provide read and write operation speeds that are comparable to DRAM [1], and due the simple structure of a memristor and the elimination of the access transistor per memory element, an ATF crossbar can offer storage densities higher than that of NAND Flash as well as other emerging memory technologies.

An inherent issue with an ATF memristive crossbar is its inability to isolate an individual memory cell from the rest of the crossbar [2]. As illustrated in Figure 1, when accessing a particular location in a crossbar (top-left in Figure 1) besides the necessary target current, there are several parasitic currents due to the partially selected devices that increase the overall power consumption of the crossbar and also impose higher current requirements for the drivers of the lines (drivers not shown in Figure 1).

The complementary resistive switch, or CRS, is an excellent candidate to mitigate these parasitic currents [3]. A CRS is formed by two anti-serially connected memristors and stores binary information as an internal configuration rather than as a resistance value. Both configurations in a CRS have a high resistance, which greatly reduces the data dependencies and parasitic current paths due to partially selected devices.

The biggest limitation for a CRS is that the readout procedure is destructive by nature and it must be followed by a restoring write operation [3]. This write amplification reduces the lifetime of the memory cells, results in excessive power consumption and can potentially lead to performance penalties as the operations involve multiple steps. Although a capacitive-based non-destructive readout procedure has been proposed for CRS-based crossbars [4], we expect this technique not to scale well, as the capacitive component of the CRS cells will decrease with the miniaturization of the cell and thus the margin to detect its state.

The destructive readout of a CRS contrasts with the non-destructive read of a memristor, in which a small sensing voltage is applied without disturbing its resistance [5]. This non-destructive read translates into higher performance, lower energy, and longer lifetime of the memory cells. On the other hand, the parasitic currents caused by the partially selected devices, result in higher energy consumption and current requirements, which ultimately limit the scalability of the memory. It seems then that the memristor behavior is preferred if we want to access a device whereas the CRS behavior is preferred if a device is not directly accessed but is partially selected.

Motivated by the contrasting but complementing behaviors of memristors and CRS devices, we present in this paper a low-power hybrid reconfigurable architecture that takes advantage of the dual CRS/memristive behavior of CRS devices [3, 6] to exploit their individual benefits and to minimize their drawbacks. Previous work exploring this dual behavior has shown that substantial energy savings can be obtained if an application exhibits strong locality of reference [7].

Our architecture for this hybrid reconfigurable resistive random-access memory, or HReRAM for short, divides the memory in pages and keeps them in either the *memristive mode* or in the *CRS mode*. Since the memristive mode is

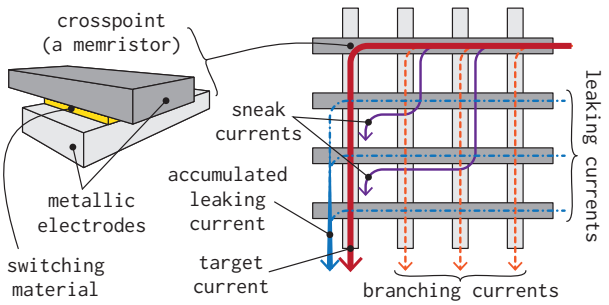

Figure 1: An access-transistor-free (ATF) memristive crossbar and its parasitic currents.

978-1-4673-9211-2/16/$31.00

faster and provides longer lifetime of the memory cells, it is used for pages that are frequently or most likely to be used. Other seldom used or unused memory pages are kept in the CRS mode and thus do not contribute as much in the energy consumption while being partially selected in a crossbar.

This is conceptually analogous to the roles of the main memory and disk in the traditional memory hierarchy: Pages in the memristive mode are akin to the main memory, pages in CRS mode to the disk, and a page fault mechanism is used to swap pages between these two modes. If we want to access a page that is in the memristive mode (main memory), we simply access it. To access a page that is in the CRS mode (disk), we first *activate* the page, which changes it from the CRS to the memristive mode, and then we access it as usual. If we later need to return a page that is in the memristive mode (main memory) back to the CRS mode (disk) we do it by the *deactivation* procedure.

Our architecture offers two key differences in this analogy with the main memory/disk case: First, there is no actual data movement between the memristive and CRS modes during the activation and deactivation procedures. The same physical region that holds a memory page can be in either the memristive mode or the CRS mode. Swapping between these two modes only involves performing an operation directly on the page, but not moving data from one region to another. This advantage over the main memory/disk case is particularly beneficial as the data movement is generally considered the main performance bottleneck. Second, the size of the 'main memory' is not fixed in our architecture. Although the total memory size (memristive+CRS) is fixed, the fraction between the memristive and CRS modes can be dynamically changed to accommodate for the memory requirements of the applications. As memories with a larger fraction in the memristive mode will consume more energy, we generally want to keep only the working set [8] of an application in the memristive mode. Dynamically accommodating to the working set size of the applications has been shown to be possible with low performance overhead [9].

We evaluate our memory organization on a 1 GB memory being accessed by a core running the SPEC CPU2006 benchmark. Our simulation results show that, on average, we can obtain energy savings of 3.6x, with respect to a memory using only the memristive mode. Additionally, the expected lifetime of the memory cells, with respect to a memory that uses only the CRS-mode, is on average 6.4x higher.

1.1 Summary of Contributions

1. We present a memory architecture that exploits the dual memristive/CRS behavior, and propose its use as a system that merges main memory and disk storage.
2. To increase energy efficiency, we propose the use of on-demand page activation, and a "System Deactivation Mechanism" (SDM) to deactivate unused pages. A small "modified page table" is used to track the mode (memristive or CRS), of each page to realize this method.
3. Furthermore, while [6] only considers the read energy, this paper considered both read and write energy for total expected energy consumption. This extension is critical: [6] claims that CRS-only memories consume more energy than memristive-only memories, but we found the opposite, as the write energy dominates the total energy.
4. Our paper proposes an endurance model to estimate the lifetime of our hybrid memory.

2. BACKGROUND

2.1 Memristive Devices

Predicted in 1971 by Leon Chua [10] and later experimentally found in 2008 by the HP Labs [11], the memristor is considered the fourth fundamental circuit element, with the other three being the resistor, the capacitor and the inductor. A memristor or memristive device is a two terminal passive device in which the resistance between its terminals can be reversibly changed between a high resistance or 'OFF' state, and a low resistance or 'ON' state by means of an external voltage applied across its terminals [12]. This change in resistance is non-volatile, and it will be kept for years after the voltage application is removed [1].

In this work we assume a metal-insulator-metal or MIM memristor in which a thin layer of a switching material (usually some type of oxide) is sandwiched between two metallic electrodes, as shown in Figure 2(a). The resistance in a MIM memristor is governed by the existence or lack of a conductive filament in the layer between the electrodes.

2.1.1 Electrical Characteristics of MIM Memristors

Figure 2(b) shows the representative bipolar I-V linear characteristics of a MIM memristor [1]. Although a memristor can be programmed with high precision within a range [13], we will focus only in two operations: the *set* and *reset* operations. To *set* (or program) the device into a memristive 'ON' or '1' state, a voltage V_{set} below a negative threshold V_{th}^{-} is applied across its terminals until a conductive filament is formed, which shorts the device and lowers its resistance to R_{on}, as shown in Figure 2(c). Similarly, to *reset* (or erase) a device into a memristive 'OFF' or '0' state, a voltage V_{reset} of opposite polarity and above a positive threshold V_{th}^{+} is applied, dissolving the filament and increasing the resistance of the device to R_{off}, as shown in Figure 2(d).

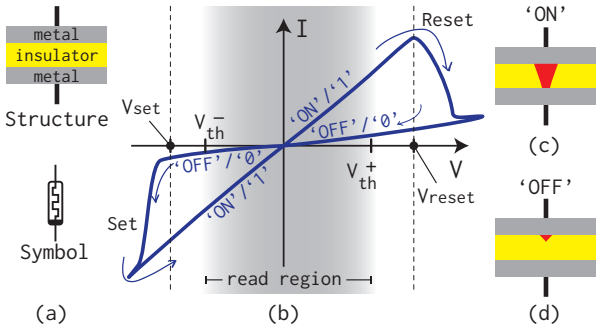

Figure 2: (a-b) Simple realization of a MIM memristor and its typical bipolar I-V linear curve. (c-d) The resistance of the device is determined by the presence or absence of a conductive filament (shown in red) between the metallic terminals.

Applying voltages between V_{th}^{-} and V_{th}^{+} (shown in gray in Figure 2(b)), will not change the resistance, thus state, of the device since the mobility of the ions forming or dissolving the conductive filament depends super-exponentially on the applied voltage [5]. This allows us to *read* without disturbing the state of the device by applying a voltage V_{read} within this gray region (called the 'read region') and measuring the resulting current [14].

2.2 Resistive Random-Access Memories

A resistive random-access memory (ReRAM) is organized as a crossbar of memristors to allow short access times that are independent of the location of the data in the crossbar [15]. Based on whether or not each memristor in the crossbar contains an *active selector* (usually a transistor), we can classify a ReRAM as *active* or *passive*.

2.2.1 Passive ReRAMs

A passive or access-transistor-free (ATF) ReRAM, is a type of "0T1R crossbar" since it has 0 transistors per 1 resistive element. In a 0T1R crossbar, the size of each memory cell is not limited by the size of the access transistor (which is often dominant) and cells as small as $4F^2$ are possible [16], where F is the minimum feature size of the crossbar. The downside of a 0T1R crossbar is that it suffers from the leakage of partially selected devices [17], which increases the energy consumption and current requirements as a function of the size of the crossbar, making larger memories impractical beyond the KB range.

2.2.2 Active ReRAMs

An active ReRAM or "1T1R crossbar" (1 transistor per 1 resistive element) allows the individual isolation of the cells, solving the problems caused by the partially selected devices. However, since the size of a 1T1R memory cell is dominated by the size of its access transistor [16], an active ReRAM is not a good candidate for applications that require high-density memories [18].

2.2.3 Quasi-passive ReRAMs

The practical approach to 1T1R and 0T1R crossbars is to use arrays of quasi-passive "1TnR crossbars" in which 1 transistor is shared by n resistive elements. To increase memory capacity, we could limit n and increase the number of 1TnR crossbars, instead of increasing n (as we would do for a 0T1R crossbar). A 1TnR crossbar has the similar benefit of having high density as that of a 0T1R crossbar, but its energy consumption and current requirements are independent of the memory size. The CMOL interface [19] allows the monolithic integration of quasi-passive crossbars on a standard CMOS process [20, 21].

A note on the similarities of 0T1R and 1TnR crossbars

The lines forming a 0T1R crossbar will eventually interface a transistor (or other driver) forming some type of 1TnR crossbar. The difference between a 0T1R crossbar and a 1TnR crossbar is that in the former we do not limit the number of cross-points per line, whereas in the latter the number of devices is limited to a target number n by design. For the purpose of discussions in this paper, we assume 0T1R and 1TnR crossbars to be both instances of ATF ReRAMs. We further assume the use of the CMOL interface, but for visual clarity we present our ideas with arrays of 1TnR crossbars.

2.3 Parasitic Currents in ATF ReRAMs

Even for 1TnR crossbars, the energy dissipated by the partially selected devices, still dominates the total energy of an operation. For instance, in a 1TnR crossbar of size 100×100 ($n = 100$), on average only about 1% of the total energy spent is consumed by the target device being accessed. This corresponds to the red *target current* in Figure 1, that either carries information about the resistance of the cross-point (during a read), or it is the current needed to write into the cross-point (during the set or reset operations). The rest 99% of the energy is dissipated by those partially selected devices. Out of this 99%, for a typical memristive R_{off}/R_{on} ratio of, say 50, about 98% of the energy is due to devices that are in the low resistance 'ON' state.

In addition to the target current, Figure 1 shows the different sources of parasitic currents in an ATF crossbar. The *branching currents* branch from the target current during read and write operations. The *leaking currents* merge into the target current during write operations and may cause issues while reading as their accumulated amount may become comparable to the target current. The *sneak currents* originate from electric charges that branch at some point from the target current and merge back with the target current at a later point. The sneak currents are the source of the *sneak-path problem* and cause issues when reading the state of a cross-point.

Although it is possible to avoid the effect of the sneak and leaking currents during read operations [22, 23, 24], the relatively high energy consumption while reading and writing on a memristive ATF crossbar still present a major challenge.

2.4 Complementary Resistive Switches

Proposed by E. Linn *et al.* [3], a *complementary resistive switch* or CRS effectively eliminates the effect of the parasitic currents described in Section 2.3. A CRS is formed by two

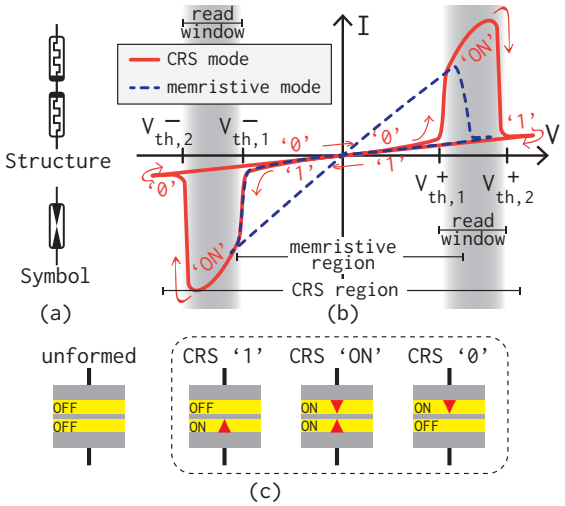

Figure 3: A complementary resistive switch. (a) its structure and symbol, (b) typical I-V curve and (c) possible memory states with their corresponding top and bottom resistances.

anti-serially connected memristors (Figure 3(a)) in which at least one of them is kept in the high resistance or 'OFF' state, providing a high resistance to the CRS regardless of the data stored on it. Recalling that most of the energy consumed by the partially selected devices comes from devices in a low resistance state, a *CRS-based crossbar* may offer substantial energy savings when compared to a *memristive-based crossbar*.

In a CRS, binary information is encoded by the position of the memristor that is in the 'OFF' state, as shown in Figure 3(c). If the top memristor is in the 'OFF' state and the bottom in the 'ON' state, the CRS is said to be in the CRS logic '1' state. Conversely, if the top device is 'ON' and the bottom device is 'OFF', the CRS is in the logic '0' state. If both devices are 'ON', the CRS is said to be 'ON', indicating a low resistance. The CRS 'ON' state is not used to represent logic. It only occurs during the 0→1 and 1→0 transitions and during the readout procedure [3]. The CRS configuration in which both devices are 'OFF' does not occur during the normal operation of the memory, but only right after fabrication when the device has not been electroformed.

The typical I-V curve of a CRS is illustrated in Figure 3(b) with a solid red line. Note that for applied voltages between $V^{-}_{th,1}$ and $V^{+}_{th,1}$, the CRS have a high resistance regardless of its logic state. Using a voltage application scheme in which the partially selected devices fall within this high resistance region can reduce considerably the energy consumption and current requirements. Negative applied voltages below $V^{-}_{th,2}$ cause the CRS to switch from the '1' to '0' states. Positive voltages above $V^{+}_{th,2}$ cause the '0' to '1' transition. Regardless of the transition (0→1 or 1→0), we call this operation the *CRS write* operation.

To read the state of a CRS, we have to apply a voltage between $V^{-}_{th,1}$ and $V^{-}_{th,2}$ (the *left read window*) and measure the resulting current. A high current indicates that the CRS stored a logic '1', whereas a low current a logic '0'. Note that if we use the *right read window* between $V^{+}_{th,1}$ and $V^{+}_{th,2}$, a high current indicates a logic '0' and a low current a logic '1'. Also note that reading a CRS could be a destructive operation since one of the two logic states will be altered after a read. As such, a read operation must be followed by a conditional restoring write operation, consuming additional energy and negatively affecting the lifetime of CRS cells.

3. MOTIVATION AND PROPOSAL

There is always a need for faster, lower-power and higher-density memory. Memristors and CRS devices are good candidates to meet this need, but as summarized in Table 1, each technology can only partially offer the necessary features. Whereas a memristor has good features for memory cells that we want to access (first four entries in Table 1), it is not good for partially selected cells (last four entries in Table 1). The CRS features, on the other hand, are excellent for partially selected cells, but not good for cells that need to be accessed. It is evident from these contrasting but complementing behaviors that a hybrid memory that exploits the benefits of both technologies is a promising solution.

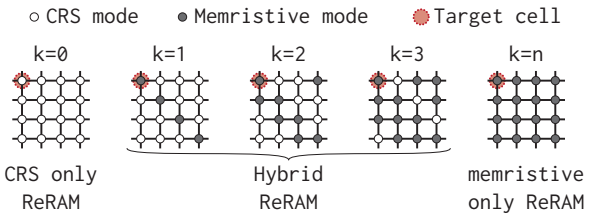

Figure 4: CRS/memristive ReRAM crossbar cases.

3.1 Memristive/CRS Dual Behavior

If instead of using the three possible states of a CRS (CRS '1', CRS 'ON' and CRS '0') shown in Figure 3(c), we restrict its operation to only two of them, e.g., CRS '1' and CRS 'ON', we obtain a behavior that is essentially the same as the memristive behavior described in Section 2.1.1. Note that choosing CRS '0' and CRS 'ON' results in the same behavior but with opposite polarity. In either case, only one of the two memristors forming the CRS is switched between its 'OFF' and 'ON' states, while the other memristor is kept in its low resistance 'ON' state. Thus, the CRS looks like a memristor with a small resistance in series.

Indeed, it has been experimentally observed for tantalum oxide-based single cell memristors, that for the same device, by simply changing the voltage application range, either a CRS or a memristive behavior can be obtained [6]. Referring to Figure 3(b), if we apply voltages in the *CRS region*, we will observe the typical CRS behavior described in Section 2.4. In these conditions, we say that the device is in the *CRS mode*. If we apply voltages in the smaller *memristive region*, a memristive behavior is observed instead, as shown in Figure 3(b) with a blue dashed line. A device operating in this region is said to be in the *memristive mode*.

3.2 A Hybrid ReRAM

To fully exploit this dual memristive/CRS behavior, we propose a memory architecture in which the frequently used data is stored in the diagonals of a crossbar. We keep these diagonals in the memristive mode and the rest of the crossbar in the CRS mode, as shown in the hybrid cases in Figure 4. By doing this, when accessing a device in a memristive diagonal (say the target cell in Figure 4) we will have all the benefits of a memristor, and all CRS benefits of the partially selected devices that are in the CRS mode.

The number k of diagonals that are kept in the memristive mode, and the probability of accessing one of these diagonals, determines the energy consumption and lifetime of the memory cells of a crossbar. In Figure 4 we show a few repre-

Table 1: Qualitative memristor/CRS comparison

Feature	Memristor	CRS
Performance	✓ High	✗ Low
Endurance	✓ High	✗ Low
Destructive read	✓ No	✗ Yes
Target cell energy consumption	✓ Low	✗ High
Other cells energy consumption	✗ High	✓ Low
Data dependencies	✗ Yes	✓ No
Current requirements	✗ High	✓ Low
Scalability	✗ Low	✓ High

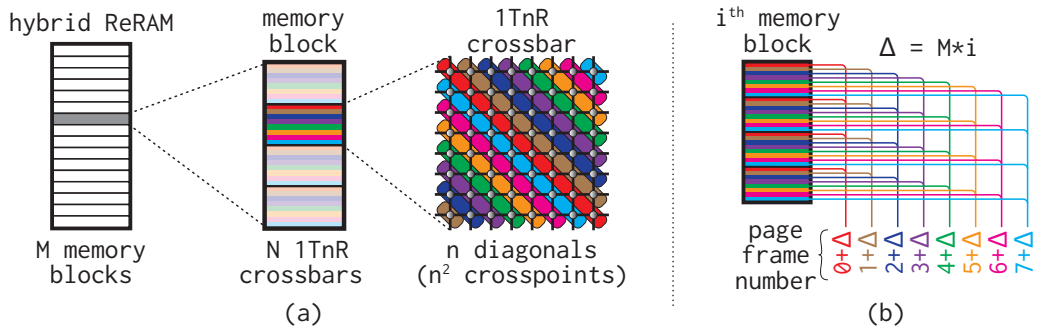

Figure 6: (a) Main memory logical division. (b) Page frame number distribution.

sentative cases in a small 4×4 crossbar ($n = 4$). In practice we expect larger crossbars in the range of $n = [10, 100]$.

The special case of $k = 0$ is called a *CRS-only ReRAM* as all the cells are in the CRS mode. Similarly, the case when $k = n$ is called a *memristive-only ReRAM*. Anything in between is considered a *hybrid ReRAM*. Our claim is that a hybrid ReRAM provides a balanced tradeoff between the energy consumption and lifetime of the memory cells.

3.2.1 Hybrid ReRAM Energy/Lifetime Performance

There are two important parameters that determine the performance of our hybrid ReRAM. The first is the locality of reference in the memory patterns of the applications. The second is the relative size of the memory requirements of the applications, with respect to the total available memory in the system.

In terms of energy consumption, the best-case scenario occurs for applications with a strong locality of reference and/or a small memory footprint. This is because we can store the applications' data in crossbars with a low k. The energy consumption in this case will be lower than the memristive-only and CRS-only cases. The worst-case scenario occurs for applications with weak locality of reference and/or large memory footprints. In this case, we will need memory regions with $k \approx n$, i.e., the memory will look like a memristive-only ReRAM, and thus will consume energy as such.

In terms of the lifetime of the memory cells, the hybrid ReRAM can only be as good as the memristive-only case, as they are both built with the same type of devices. However, the additional memory management of the hybrid case, which is not present in the memristive-only case, reduces its lifetime, as it involves additional writes. Conversely, the hybrid ReRAM is expected to have a longer lifetime than the CRS-only case, providing we frequently access memory cells that are in the memristive mode. We can conclude that with a low overhead memory management and a strong locality of reference in the applications, the lifetime of the hybrid case should be between the CRS-only and memristive-only cases, but closer to the memristive-only case.

4. MEMORY ORGANIZATION

4.1 Memory Hierarchy

The faster-than-Flash write and read speeds of memristive crossbars allow us to place them closer to the CPU in the memory hierarchy, effectively closing the gap between the main memory (DRAM-based) and disk storage [25]. It has been shown that phase change memory (PCM) coupled with a small DRAM buffer can be used to provide DRAM-like latencies with the capacity benefits of PCM [26]. Likewise, we envision our hybrid ReRAM as an aggressive system located after a DRAM buffer, but in which the main memory is merged with the disk storage, as shown in Figure 5. Depending on the speed (relative to DRAM) of the hybrid ReRAM and the higher endurance of future memristive devices, the DRAM buffer can be entirely omitted to form a fully non-volatile memory system. In our hybrid ReRAM, frequently accessed data is kept in the memristive mode (playing the role of main memory), while seldom used or unused data is kept in the CRS mode (playing the role of disk). Upon memory requests that are in the CRS mode (disk), the block or page containing the requested address is simply changed from the CRS to the memristive mode. Unlike the main memory/disk case, this mode change does not incur in any data traffic.

4.2 Memory Division

Figure 6(a) shows the logical division of our hybrid ReRAM. The main memory is divided into *M memory blocks*. Each memory block is further divided into N 1TnR crossbars, which in turn consist of *n diagonals* each, with n cross-points per diagonal. Each diagonal is associated with a unique color or index.

With this division, a memory page, in its most common sense, is split and spread across a memory block, as shown in the color-coded Figure 6(b). Specifically, a given page in main memory will be formed by the N diagonals of the same color located in the same memory block, and can be uniquely addressed with its memory block index and color. Typical 4 KB pages could be obtained with 1TnR crossbars of 64×64 cross-points ($n = 64$) and memory blocks of at least $N = 512$.

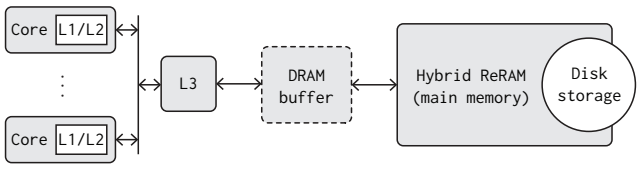

Figure 5: Generic memory hierarchy using our hybrid ReRAM as main memory and storage.

4.3 Role of the OS

Each page can be in one of two modes: the memristive (or *activated*) mode or in the CRS (or *deactivated*) mode. An OS-maintained *modified page table* (MPT) keeps track of the mode of each page, in the same way it keeps track on the swapped status of a page in a modern OS. We introduce an additional but smaller table of M entries (one entry per memory block), called the kMap, that is used by the OS to keep track of the number k of diagonals that are in the memristive mode (a number between 0 and n). Since the energy consumption of a 1TnR crossbar is directly proportional to the number of memristive devices in a low resistance (ON) state, one could argue that the value k of a memory block, i.e., the number of activated diagonals, is a good estimation of the energy such block will consume if accessed. In this way, upon memory allocation requests, and based on the optimization goal, the OS should provide memory regions with low k to reduce the overall energy consumption of the memory.

4.3.1 Dynamic Activation and Deactivation of Pages

Activating and deactivating pages, i.e., changing between the memristive and the CRS modes, hurts the lifetime of the memory cells and consumes energy. As such, their use should be minimized. In general, the OS allocates memory in a mixture of pages in the memristive and CRS modes. The activation is triggered on demand, only when an application attempts to access a deactivated page. In such case, the page is first activated to the memristive mode before being accessed.

The activation mechanism will eventually convert most pages into the memristive mode so that the memory will look like a memristive-only memory. To minimize unnecessary waste of energy, an OS process periodically checks all the activated pages, and deactivates unused ones back into the CRS mode. We call this memory-wide process the *system deactivation mechanism* or SDM. The used or unused status of each page is determined by its *used bit* in the MPT. This bit is set every time a page is accessed and cleared for all the pages every time the SDM is executed.

5. PERFORMANCE METRICS

We use two metrics to evaluate the performance of our hybrid memory: (1) its energy consumption and (2) its aging factor. The aging factor is the inverse of the expected lifetime of the memory cells. We use these metrics to compare our hybrid architecture against two baselines: a memristive-only ReRAM and a CRS-only ReRAM. The former keeps all its memory cells in the memristive mode, the latter keeps them in the CRS mode. In both cases the activation and deactivation mechanisms will never be triggered.

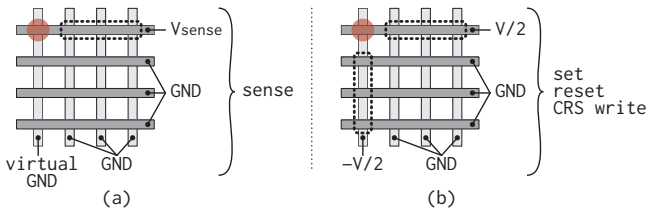

Figure 7: Voltage application schemes.

We expect the memristive-only case to have the smallest aging factor but the highest energy consumption. The CRS-only case is expected to have the highest aging factor, but generally, a lower energy consumption.

6. ENERGY MODEL

The energy consumption to access a device in a 1TnR crossbar depends on the voltage application scheme, and on the operation. We assume a *virtual ground* scheme [24] for reading and a $V/2$ scheme for the set, reset and CRS write operations, in which $V/2$ and $-V/2$ voltages are applied on the target memristor's terminals while other lines are grounded. Figure 7 summarizes these schemes when accessing the top-left device of a small 4×4 crossbar. Figure 7 also highlights with a dashed box the partially selected devices that contribute with parasitic currents in each case.

To analyze the average energy consumption of a 1TnR crossbar and make a fair comparison between the memristive-only, CRS-only and hybrid cases, we define the set of relevant parameters in Table 2. The *memristive hit rate h* is the percentage of memory accesses that address memory cells in the memristive mode. The rest $1-h$ is for accesses that address cells in the CRS mode. We use parameter m to express the fraction of the total memory that is in the memristive mode, and thus $1-m$ for the fraction in the CRS mode. Parameter p represents the probability of having data stored in the memory being logic '1'. The energy ε is defined as the read energy consumed by a single memristor that is in the low resistance ON state. The set and reset energies for a memristor as well as the write energy of a CRS, are expressed as linear functions of ε, with multiples of S, R and C, respectively. With the assumption of devices having linear behaviors and by denoting the energy consumption of a device in the ON state E_{on}, the energy in the OFF state E_{off} can be approximated by dividing E_{on} by r, where r is the $R_{\text{off}}/R_{\text{on}}$ ratio of a memristor. We use these parameters to estimate the energy consumption of a 1TnR crossbar, not including that of the CMOS subsystem.

6.1 Average Read/Write Energy Consumption

With the parameters of Table 2, we proceed to define the average read and write energy for each one of the study cases: the memristive-only, CRS-only, and hybrid cases. Being an average, we expect these energy numbers to be the representative sustained behavior of each case.

The total energy consumed by a 1TnR crossbar when sensing[1] the state of a memory cell in the **memristive** mode is defined as the **average sense energy**, and it is given by:

$$\overline{\mathrm{E_s}}(m) = \left[p + \frac{1-p}{r} + \left[mp + \frac{1-mp}{r}\right]n'\right]\varepsilon. \qquad (1)$$

The total energy when writing[2] into a cell in either **memristive** or **CRS** mode is defined as the **average write energy**

[1] We assume the 'sensing' operation to be exclusive to cells in the memristive mode.

[2] 'Writing' here is any generic operation whose purpose is to change the state of a device.

Table 2: Description of parameters for power analysis.

Parameter	Description
h	Percentage of memory accesses falling in memristive mode
$1-h$	Percentage of memory accesses falling in CRS mode
m	Memory fraction in memristive mode
$1-m$	Memory fraction in CRS mode
p	Percentage of data in the memory being logic 1
$1-p$	Percentage of data in the memory being logic 0
w	Percentage of write operations
$(1-w)$	Percentage of read operations
d	Probability of deactivating a page
$n \times n$	Size of a 1TnR crossbar
n'	Number of partially selected devices: $n-1$
k	Number of devices in memristive mode per line
r	R_{off}/R_{on} ratio of a memristor
ε	Read energy of an ON memristor (our base energy)
ε/r	Read energy of an OFF memristor
$\varepsilon \times S$	Set energy of a memristor (OFF $\rightarrow$ ON transition)
$\varepsilon \times R$	Reset energy of a memristor (ON $\rightarrow$ OFF transition)
$\varepsilon \times C$	Write energy of a CRS ($1 \rightarrow 0$ or $0 \rightarrow 1$ transitions)

and it is given by:

$$\overline{\mathrm{E_w}}(x,y,m) = \left[y + \frac{1-y}{r} + \left[mp + \frac{1-mp}{r} \right] \frac{n'}{2} \right] \varepsilon x, \quad (2)$$

where x is either S, R or C depending on the operation being a set, reset or CRS write, respectively, and y is the probability of the target cell being in a low resistance state.

Since in general, the energy to set and reset a device is different, i.e., $S \neq R$, using Equation (2), we merge the effect of a set and a reset into the **average set/reset energy**, which we define as:

$$\overline{\mathrm{E_w^{SR}}}(m) = p\overline{\mathrm{E_w}}(S,p,m) + (1-p)\overline{\mathrm{E_w}}(R,p,m). \quad (3)$$

We further define the **average activate+sense energy** as the energy consumed by an operation that activates a cell (from the CRS to the memristive mode), and it is given by:

$$\overline{\mathrm{E_{a+s}}} = \overline{\mathrm{E_w}}(S,0,m) + \overline{\mathrm{E_s}}(m) + (1-p)\overline{\mathrm{E_w}}(C,0,m), \quad (4)$$

and the **average deactivation energy**, which deactivates a cell (from the memristive to CRS mode), as:

$$\overline{\mathrm{E_d}} = \overline{\mathrm{E_s}}(m) + p\overline{\mathrm{E_w}}(R,1,m) + (1-p)\overline{\mathrm{E_w}}(C,0,m). \quad (5)$$

With Equations (1-4), we can compute the **average read energy** and **average write energy** for a memristive-only (M), CRS-only (C) and hybrid cases (H), respectively, as:

$$\overline{\mathrm{E_M^{read}}} = \overline{\mathrm{E_s}}(1),$$
$$\overline{\mathrm{E_M^{write}}} = \overline{\mathrm{E_w^{SR}}}(1),$$
$$\overline{\mathrm{E_C^{read}}} = \overline{\mathrm{E_w}}(S,0,0) + \overline{\mathrm{E_s}}(0) + p\overline{\mathrm{E_w}}(R,1,0),$$
$$\overline{\mathrm{E_C^{write}}} = \overline{\mathrm{E_w}}(C,0,0),$$
$$\overline{\mathrm{E_H^{read}}} = h\overline{\mathrm{E_s}}(m) + (1-h)\overline{\mathrm{E_{a+s}}},$$
$$\overline{\mathrm{E_H^{write}}} = h\overline{\mathrm{E_w^{SR}}}(m) + (1-h)\left[\overline{\mathrm{E_w}}(C,0,m) + p\overline{\mathrm{E_w}}(S,0,m)\right].$$

Finally, a representative **total average energy** for each case can be obtained as a weighted combination of the read and write components:

$$\overline{\mathrm{E_M^{total}}} = w\overline{\mathrm{E_M^{write}}} + (1-w)\overline{\mathrm{E_M^{read}}}, \quad (6)$$
$$\overline{\mathrm{E_C^{total}}} = w\overline{\mathrm{E_C^{write}}} + (1-w)\overline{\mathrm{E_C^{read}}}, \quad (7)$$
$$\overline{\mathrm{E_H^{total}}} = w\overline{\mathrm{E_H^{write}}} + (1-w)\overline{\mathrm{E_H^{read}}} + d\overline{\mathrm{E_d}}. \quad (8)$$

Where d is the deactivation probability. Note that the activation energy is included in the $\mathrm{E_{a+s}}$ term.

7. AGING FACTOR MODEL

The endurance is the number of programming or write cycles that can be performed before a device becomes unreliable. A higher endurance translates into a longer lifetime, which is equivalent to a smaller aging factor. To model the aging factor and enable a fair comparison between the memristive-only, CRS-only, and hybrid cases, we need information of the relative endurance for memristors and CRS devices. In particular, we need to determine what we call the *relative endurance factor* or REF, which captures how many more times a cell in the memristive mode can be programmed, compared to a cell in the CRS mode.

The aging factor of a CRS-based memory depends itself on two factors: the REF, and the read/write ratio of the applications. A read counts towards the programming quota, as a CRS read is destructive, and must be followed by a restoring write. For write-heavy workloads, we expect the CRS-only case to have an aging factor of approximately REF times higher than a memristive-only case. For read-heavy workloads, the aging factor of the CRS-only can be significantly higher than the REF. In general, the CRS aging factor should be greater than the REF.

The aging factor for each of the cases (memristive-only, CRS-only or hybrid), is computed by counting the *effective*

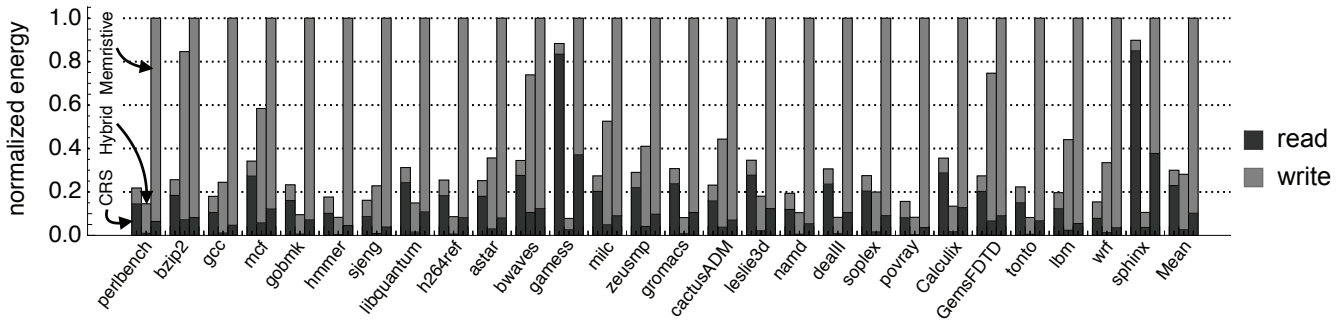

Figure 8: Normalized energy consumption (w.r.t. the memristive-only case).

number of writes (actual writes + restoring writes) of every byte in the memory, and then we add the counts. We report on the normalized total count for each case with respect to the CRS-only case (the worst case).

Whereas the individual endurance of the memristive and CRS devices have been reported elsewhere [27, 28], there are limited data on the REF of the same device stack. The experimental data reported on [29] and in the supplementary material of [6], show that, for the same device stack, the memristive mode can be cycled about 10 times more than the same device in the CRS mode, i.e., REF ≈ 10.

Table 3 contains the average effective number of writes for each case per accessed byte. We assume the probability of having logic 1 and logic 0 in the memory to be the same ($p = 0.5$). For the memristive-only and CRS-only cases, out of the four possible combinations (writing a 0 or 1 into a cell with a 0 or 1 stored), only two results in switching the value of the cell. Note that the hybrid case has the same values as the memristive-only case for reads and writes. Activating and deactivating, however, are considerably expensive, and they are not present in the memristive-only and CRS-only cases.

8. METHODOLOGY DESCRIPTION

We study the performance of our hybrid ReRAM with the SPEC CPU2006 benchmark. We use Intel's Pin tool [30] to capture the memory trace of each benchmark, which is then fed to an in-house memory simulator. The memory trace is produced on a Linux machine with a 3.50 GHz Intel Xeon CPU with 128 GB of main memory. The simulator runs in the same machine. The virtual to physical address translation is performed on the simulator via the Linux's pagemap interface [31]. The simulated system consists in:

- a core running the benchmarks (the memory trace),
- a 32 KB, 8-way, LRU, 64 B line size L1 cache,
- a 256 KB, 8-way, LRU, 64 B line size L2 cache,
- a 1 GB ReRAM main memory,
- a disk with unlimited size for storage.

Our simulator runs three instances of the system in which the 1 GB ReRAM main memory is either a memristive-only ReRAM, a CRS-only ReRAM or a hybrid ReRAM. The main memory in all cases is organized in pages of 4 KB with $M = 4096$, $N = 512$ and $n = 64$. All three instances see the same memory trace and place the pages in the same location for a fair comparison.

To compute the aging factor, for all three cases we keep a counter per byte of the effective number of writes using Table 3. To estimate the energy consumption, we keep for each of the $M = 4096$ memory blocks, a counter to track the number of bytes that are read (*RBytes*), written (*WBytes*) and deactivated (*DBytes*). For the hybrid case we keep an additional global counter for the number memory accesses that fall in the memristive mode (*MAccesses*), and another for the total number of memory accesses (*TAccesses*).

For every 1 billion memory instructions, the *analysis routine* is executed, in which we compute:

$$\text{Read Energy (X case)} = \sum_{i}^{\text{All blocks}} \overline{\text{E}_{\text{X}}^{\text{read}}} \times \mathit{RBytes}(\text{X})_i,$$

$$\text{Write Energy (X case)} = \sum_{i}^{\text{All blocks}} \overline{\text{E}_{\text{X}}^{\text{write}}} \times \mathit{WBytes}(\text{X})_i,$$

where X stands for M, C or H for the memristive-only, CRS-only or hybrid cases, respectively. In addition, for the hybrid case we compute:

$$\text{Deactivation Energy} = \sum_{i}^{\text{All blocks}} \overline{\text{E}_{\text{d}}} \times \mathit{DBytes}(\text{H})_i.$$

The hybrid case requires the computation of two parameters: the memristive hit rate h and a memristive fraction m_i per block. h is simply $\mathit{MAccesses}/\mathit{TAccesses}$, whereas $m_i = k_i/n$, where k_i is the k value of the ith memory block (read from the kMap). After running the analysis routine, all the counters are reset to zero and the simulation continues.

If the system deactivation mechanism (SDM) for the hybrid case is enabled, it is triggered with a periodicity given by the *deactivation period*.

Table 3: Effective number of writes

Operation	Memristive	CRS	Hybrid
Read	0	1	0
Write	1/2	REF/2	1/2
Activation	–	–	(REF + 1)/2
Deactivation	–	–	(REF + 1)/2

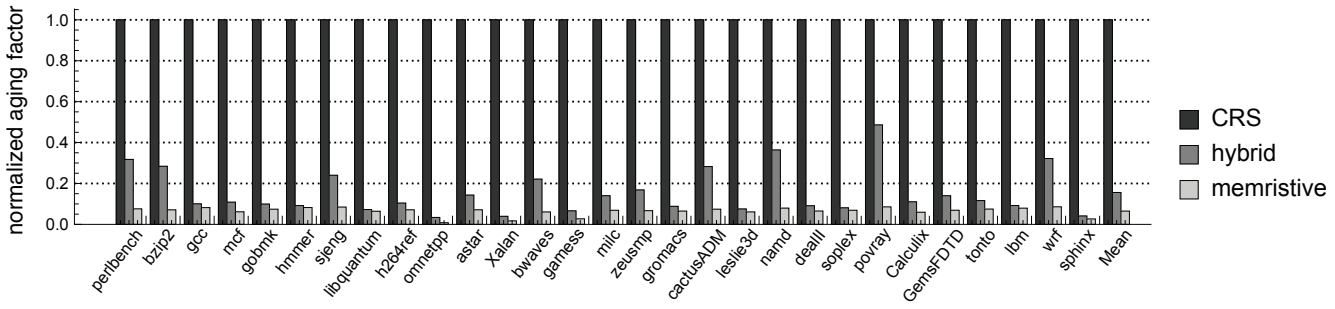

Figure 9: Normalized aging factor (w.r.t. the CRS-only case).

9. RESULTS AND ANALYSIS

9.1 Parameters

Our simulation parameters are based on [7]. A $V_{\text{read}} = 0.1\,\text{V}$, a read time t_{read} of 20 ns, set voltage V_{set} of 1.0 V and set time t_{set} of 2 ns. Since the energy ε can be computed as $\varepsilon = t_{\text{read}} \cdot V_{\text{read}}^2/\text{R}_{\text{on}}$, and the set energy can be approximated as $\varepsilon \times S = t_{\text{set}} \cdot V_{\text{set}}^2/\text{R}_{\text{on}}$, we found that $S \approx 10$. We chose $R = 80$ as the ITRS reports reset energies that are 8x larger than the set energies. We set $C = 90$ as it involves a set and reset. We assumed $p = 0.5$, a $r = \text{R}_{\text{off}}/\text{R}_{\text{on}}$ of 50, and 1TnR crossbars of size 64×64, i.e., $n = 64$. Based on [29, 6] we chose REF $= 10$.

9.2 Energy Consumption

Figure 8 summarizes the normalized energy consumption (with respect to the memristive-only case) after 5 billion instructions for a subset of the SPEC CPU2006 benchmarks. In these results, we disabled the SDM, i.e., the memory does not perform deactivations for unused pages. The first, second and third set of bars in Figure 8 correspond to the CRS-only, hybrid and memristive-only cases, respectively. The benchmarks in Figure 8 were chosen as they have a similar read/write ratio in the first 5 billion instructions. Benchmarks `omnetpp` and `xalancbmk` will be discussed in Section 9.2.1.

Note the difference in the read and write energy distribution in each case. For the CRS-only case, the read operation dominates the total energy consumption. This is consistent with the observation that the CRS read is a more complex operation that requires a write, a sensing, and a conditional restoring write. Conversely, the energy in the memristive-only case is dominated by the write operation, with an average of about 90% of the total energy. This is due to the high leakage currents from the partially selected devices in the low resistance ON state. Finally, in the hybrid case (our proposed architecture), the energy is also dominated by the write operation, but the total energy is significantly lower, with an average of less than 30% of the energy in memristive-only case, and very similar to the CRS-only case.

Note that, whereas the total CRS-only energy in each benchmark is close to the 30% average, the hybrid case has a much wider variation. The reason is that the energy consumed by the hybrid memory is directly proportional to the memristive fraction of the memory. A higher memristive fraction allows us to have a larger memory (more activated pages), but consumes more energy (it becomes similar to the memristive-only case). For instance, for the `bzip2` benchmark, that has a stable memory footprint of 856 MB (85% of the total memory) in the first part of its execution [32], consumes in our hybrid memory 85% of the energy of the memristive-only case. The `gobmk` benchmark, that has a relatively stable footprint that does not exceed 30 MB [32], consumes in our hybrid memory about 9% of the memristive-only energy.

9.2.1 *Read-heavy Phases*

Figure 10 shows the evolution of the first 50 billion instructions of the read, write and total energies in the `omnetpp` benchmark (not shown in Figure 8). `omnetpp` starts with a phase of 10 billion instructions of heavier read activity, and then switches to a more typical read/write ratio phase, in which the write operation dominates. Note that in the first phase, the memristive-only memory consumes less energy than the CRS-only case (contrary to the behavior observed in Figure 8). After that, the behavior reverses. Also note that in both phases, our hybrid memory consumes the smallest energy, with about 15%-22% of the memristive-only energy. For the `xalancbmk` benchmark, we observed a trend similar to those of Figure 10.

9.3 Aging Factor

Figure 9 shows the normalized aging factor (with respect to the CRS-only case) for the complete suite of SPEC CPU2006 benchmarks after running the first 5 billion instructions with the SDM disabled. As discussed in Section 7, we found the relative aging factor of the memristive-only memory to be less that 10% ($1/\text{REF}$) of the CRS-only case, with an average value of 6.5%.

Our hybrid memory also performs very well, with an average relative aging factor of 15.6%, i.e., an expected lifetime of 6.4x longer than the CRS-only memory. The noticeable poor performance in some of the benchmarks, `povray` or `namd` in particular, we believe, is mainly due to the fact that the benchmarks have a small memory footprint (9 MB and 53 MB for `povray` and `namd`, respectively). The hybrid memory case was simulated with initially all the cells in the CRS mode, so every memristive miss causes the activation of a complete page. This initial penalty in endurance cannot be amortized if, for example, the application initially touches all its memory, and then accesses just a small memory region. This seems to be the case for the `povray` benchmark that has a working set size (WSS) of only 0.4 MB [33].

9.4 SDM Evaluation

We now proceed to enable the System Deactivation Mechanism (SDM) to study its effect on the energy consumption and on the aging factor. The SDM is exclusive to the hybrid case, and it consumes energy and contributes to the aging of the cells. The SDM is particularly useful when the applications exhibit small WSSs, but larger memory footprints. One of such applications is the `dealII` benchmark, that has a monotonically increasing memory footprint from few MB up to 564 MB [32], but with a WSS that follows the shape of the memory footprint with short memory peaks, and then stays relatively low in the 10 MB-100 MB range [33].

In Figure 11 we show the normalized lifetime (inverse of the aging factor) as well as the normalized energy savings (inverse of the energy consumption), as a function of the deactivation period in the `dealII` benchmark. The lifetime and energy savings are normalized with respect to the CRS-only and memristive-only cases, respectively. Both metrics are evaluated after executing the first 200 billion instructions of `dealII`. As references, we include the values of both metrics for the CRS-only and memristive-only cases.

With no deactivations (ND), shown in the leftmost points in Figure 11, our hybrid memory can achieve a relative lifetime of almost 24x with an energy savings of about 3.3x. With a deactivation period of 10 billion instructions (10B), the energy savings increase to 4.5x (an increment of 36% from the ND case), but the lifetime reduces to 11.5x (a reduction of 50% from the ND case). With a deactivation period of 5 billions instructions (5B), we obtain 5.8x in energy savings (increment of 29% from the 10B case) and 10.1x in lifetime (reduction of only 12% from the 10B case). For any value after this point (deactivation periods of 200 million instructions or less), even though the energy savings can be increased to 6.6x, the expected lifetime of <3.4x makes the SDM unattractive. The optimal deactivation period of a few billion instructions observed in Figure 11, is consistent with the literature in which low-overhead memory managing OS routines are triggered every 250 ms [9].

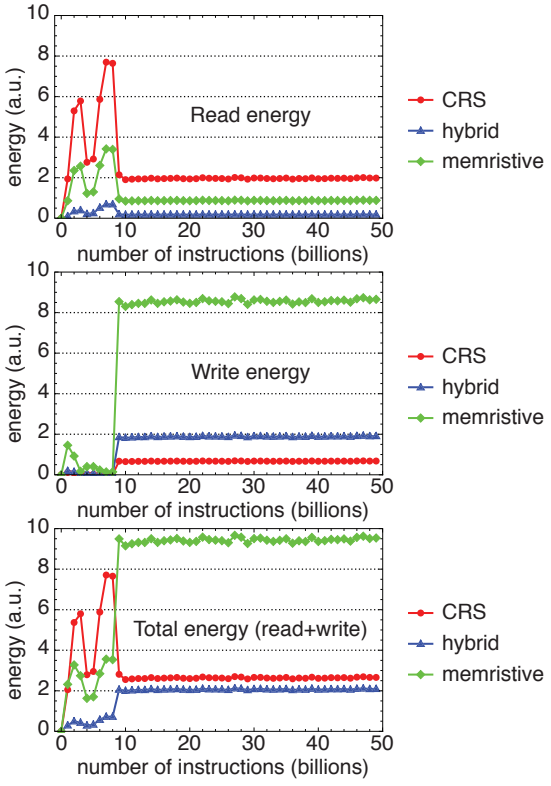

Figure 10: `omnetpp` read, write and total energy consumption for the first 50 billion instructions. The energy is in arbitrary units, but in the same scale in all cases.

10. RELATED WORK

A similar work, but exploiting the characteristics of phase-change memory (PCM) to store multiple bits per cell, is in M. Qureshi *et al.* [9]. In their work, a reconfigurable or "morphable" memory dynamically configures blocks in either a high capacity or low capacity mode. They present a solution to the tradeoff between the memory capacity and performance in PCM. Their solution is orthogonal to ours and can in fact be incorporated into our system if we also exploit the multi-level characteristics of the memristors for storing multiple bits per memristor [13].

In this paper we did not consider the wear of the memory blocks in the page placement policy, but only the memristive fraction of the blocks, which is an indicator of the energy consumption of the block. In some conditions, this could lead to a non-uniform wearing of the memory blocks. In L. Ramos *et al.* [34] a hybrid PCM/DRAM system, similar to [26], but in which a memory controller ranks pages according to their access frequency and write intensity, and migrates the top-ranked pages to DRAM, effectively improves the lifetime of the PCM cells. Our hybrid memory could leverage this

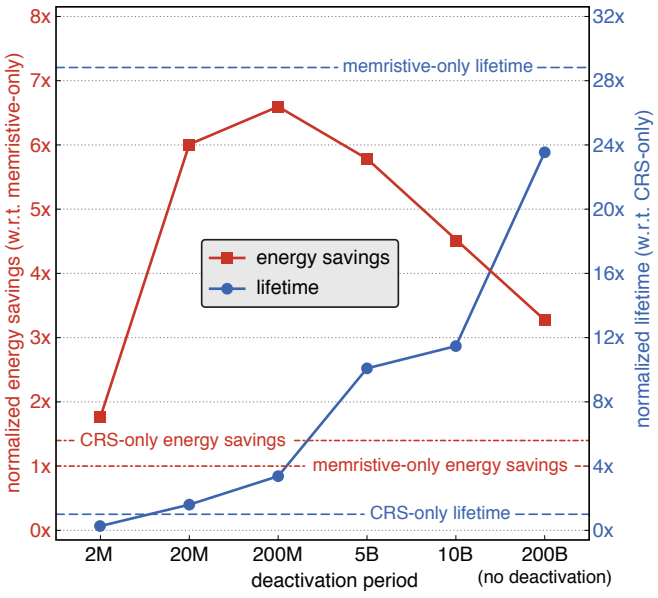

Figure 11: `dealII` lifetime-energy savings tradeoff of the hybrid ReRAM at different deactivation periods.

memory controller if we use a DRAM buffer, but it could also be used to detect hot pages, and store them in a higher endurance, small 1T1R ReRAM, which effectively works as a buffer of our memory.

11. CONCLUDING REMARKS

In this paper we propose a hybrid reconfigurable architecture and memory division for resistive random-access memories based on memristive crossbars. In particular we propose the use of the observed dual behavior of a complementary resistive switch (CRS), in which, by changing the voltage application range, either a memristive behavior or a CRS behavior can be obtained.

Not only our hybrid architecture suppresses the various sources of leakage currents caused by the partially selected devices (as we keep them in the CRS mode) but also enjoys all the benefits of accessing a cell in the memristive mode (including high performance, high endurance, and non-destructive read). We enable the coexistence of both modes by storing the frequently accessed devices in the diagonals of the crossbars configured in the memristive mode, and the other, less frequently used devices, in the CRS mode.

By dynamically changing the fraction of memristive devices in our hybrid memory, we can effectively accommodate the memory requirement of the applications. In all cases we can offer an energy consumption that is, in the worst case, the same as a memory that only uses the memristive-only mode, but in the typical case, much smaller. We can provide a lifetime longer than that of a memory system using the CRS-only mode. To evaluate the performance of our architecture, we simulated a 1 GB hybrid ReRAM with the SPEC CPU2006 benchmark and compared it to a CRS-only ReRAM and memristive-only ReRAM of the same capacity. We found that, compared to the memristive-only case, our architecture achieves an average energy saving of 3.6x and up to 10x for some benchmarks. In addition, our memory can achieve an average lifetime of 6.4x longer than that of a CRS-only memory.

We expect these numbers to get better for memories with a higher capacity. As such, we envision our hybrid ReRAM in a computer system in which its *main memory* consists of cells in the memristive mode, and the other memory cells in the CRS mode playing the role of *disk storage*. The biggest differences between our system and the typical main memory/disk case are:

1. The "main memory" size is not fixed, but variable, and can be adjusted to accommodate the application's requirements. A greater main memory size, however, typically incurs higher energy consumption.
2. "Fetching something from disk" does not involve any movement of data in our system. We just need to activate the page we need to "fetch" and possibly deactivate another page. The activation and deactivation are operations performed in place, so no data movement is needed. This property naturally opens the possibility of execute-in-place (XIP).

We believe that such hybrid non-volatile memory, covering the roles of the main working memory and disk storage, will play a key role in future highly integrated computer systems.

Acknowledgments

This work was supported by the Air Force Office of Scientific Research under the MURI grant FA9550-12-1-0038, and the Industrial Technology Research Institute (ITRI).

12. REFERENCES

[1] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[2] A. Ghofrani, M. A. Lastras-Montaño, and K.-T. T. Cheng, "Toward Large-Scale Access-Transistor-Free Memristive Crossbars," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, 2015.

[3] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature materials*, vol. 9, no. 5, pp. 403–406, 2010.

[4] S. Tappertzhofen, E. Linn, L. Nielen, R. Rosezin, F. Lentz, R. Bruchhaus, I. Valov, U. Böttger, and R. Waser, "Capacity based nondestructive readout for complementary resistive switches," *Nanotechnology*, vol. 22, no. 39, p. 395203, 2011.

[5] D. B. Strukov and R. S. Williams, "Exponential ionic drift: fast switching and low volatility of thin-film memristors," *Applied Physics A*, vol. 94, no. 3, pp. 515–519, 2009.

[6] Y. Yang, P. Sheridan, and W. Lu, "Complementary resistive switching in tantalum oxide-based resistive memory devices," *Applied Physics Letters*, vol. 100, no. 20, p. 203112, 2012.

[7] M. A. Lastras-Montaño, A. Ghofrani, and K.-T. Cheng, "HReRAM: A Hybrid Reconfigurable Resistive Random-Access Memory," *Proceedings Design, Automation, and Test in Europe (DATE), IEEE*, 2015.

[8] P. J. Denning, "The working set model for program behavior," *Communications of the ACM*, vol. 11, no. 5, pp. 323–333, 1968.

[9] M. K. Qureshi, M. M. Franceschini, L. A. Lastras-Montaño, and J. P. Karidis, "Morphable Memory System: A Robust Architecture for Exploiting Multi-level Phase Change Memories," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, (New York, NY, USA), pp. 153–162, ACM, 2010.

[10] L. O. Chua, "Memristor-The missing circuit element," *Circuit Theory, IEEE Transactions on*, vol. 18, no. 5, pp. 507–519, 1971.

[11] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[12] D. B. Strukov and H. Kohlstedt, "Resistive switching phenomena in thin films: Materials, devices, and applications," *MRS bulletin*, vol. 37, no. 02, pp. 108–114, 2012.

[13] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.

[14] A. Ghofrani, M. A. Lastras-Montaño, and K.-T. T. Cheng, "Towards data reliable crossbar-based memristive memories," in *Test Conference (ITC), 2013 IEEE International*, 2013.

[15] M. A. Lastras-Montaño, A. Ghofrani, and K.-T. T. Cheng, "Architecting Energy Efficient Crossbar-based Memristive Random Access Memories," in *ACM/IEEE Internation Symposium on Nano-scale Architectures (NANOARCH'15)*, 2015.

[16] "The International Technology Roadmap for Semiconductors (ITRS), System Drivers, 2013, http://www.itrs.net/,"

[17] A. Ghofrani, M. A. Lastras-Montaño, S. Gaba, M. Payand, W. Lu, L. Theogarajan, and K.-T. T. Cheng, "A Low-Power Variation-Aware Adaptive Write Scheme for Access-Transistor-Free Memristive Memory," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, 2015.

[18] A. Rahimi, A. Ghofrani, M. A. Lastras-Montaño, K.-T. T. Cheng, L. Benini, and R. K. Gupta, "Energy-efficient GPGPU architectures via collaborative compilation and memristive memory-based computing," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, 2014.

[19] K. K. Likharev and D. B. Strukov, "CMOL: Devices, circuits, and architectures," in *Introducing Molecular Electronics*, pp. 447–477, Springer, 2005.

[20] J. Rofeh, A. Sodhi, M. Payvand, M. A. Lastras-Montaño, A. Ghofrani, A. Madhavan, S. Yemenicioglu, K.-T. T. Cheng, and L. Theogarajan, "Vertical Integration of Memristors onto Foundry CMOS Dies using Wafer-Scale Integration," in *IEEE Electronic Components and Technology Conference (ECTC'15)*, 2015.

[21] K.-T. Cheng and D. B. Strukov, "3D CMOS-Memristor Hybrid Circuits: Devices, Integration, Architecture, and Applications," in *International Symposium on Physical Design (ISPD)*, IEEE, 2012.

[22] J. Mustafa and R. Waser, "A novel reference scheme for reading passive resistive crossbar memories," *Nanotechnology, IEEE Transactions on*, vol. 5, no. 6, pp. 687–691, 2006.

[23] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, 2012.

[24] R. Luyken and F. Hofmann, "Concepts for hybrid CMOS-molecular non-volatile memories," *Nanotechnology*, vol. 14, no. 2, p. 273, 2003.

[25] R. F. Freitas and W. W. Wilcke, "Storage-class memory: The next storage system technology," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 439–447, 2008.

[26] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 24–33, 2009.

[27] M.-J. Lee, C. B. Lee, D. Lee, S. R. Lee, M. Chang, J. H. Hur, Y.-B. Kim, C.-J. Kim, D. H. Seo, S. Seo, U.-I. Chung, I.-K. Yoo, and K. Kim, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures," *Nature materials*, vol. 10, no. 8, pp. 625–630, 2011.

[28] S. Schmelzer, E. Linn, U. Bottger, and R. Waser, "Uniform complementary resistive switching in tantalum oxide using current sweeps," *Electron Device Letters, IEEE*, vol. 34, no. 1, pp. 114–116, 2013.

[29] F. Nardi, S. Balatti, S. Larentis, and D. Ielmini, "Complementary switching in metal oxides: Toward diode-less crossbar RRAMs," in *Electron Devices Meeting (IEDM), 2011 IEEE International*, pp. 31–1, IEEE, 2011.

[30] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," in *ACM Sigplan Notices*, vol. 40, pp. 190–200, ACM, 2005.

[31] "pagemap, from the userspace perspective." https://www.kernel.org/doc/Documentation/vm/pagemap.txt.

[32] J. L. Henning, "SPEC CPU2006 memory footprint," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 1, pp. 84–89, 2007.

[33] D. Gove, "CPU2006 working set size," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 1, pp. 90–96, 2007.

[34] L. E. Ramos, E. Gorbatov, and R. Bianchini, "Page placement in hybrid memory systems," in *Proceedings of the international conference on Supercomputing*, pp. 85–95, ACM, 2011.