Accepted Manuscript

Adaptive programming in multi-level cell ReRAM

Misbah Ramadan, Nicolás Wainstein, Ran Ginosar, Shahar Kvatinsky

PII: S0026-2692(18)30833-4

DOI: https://doi.org/10.1016/j.mejo.2019.06.004

Reference: MEJ 4570

To appear in: *Microelectronics Journal*

Received Date: 29 October 2018

Revised Date: 15 April 2019

Accepted Date: 10 June 2019

Please cite this article as: M. Ramadan, Nicolá. Wainstein, R. Ginosar, S. Kvatinsky, Adaptive programming in multi-level cell ReRAM, *Microelectronics Journal* (2019), doi: https://doi.org/10.1016/j.mejo.2019.06.004.

Adaptive Programming in Multi-Level Cell ReRAM

Misbah Ramadan [a,1,*], Nicolás Wainstein[a,*], Ran Ginosar[a], Shahar Kvatinsky[a]

[a]*Andrew and Erna Viterbi Faculty of Electrical Engineering, Technion –- Israel Institute of Technology, Haifa, Israel, 3200003.*

Abstract

Resistive memory (ReRAM) is an attractive technology to replace Flash technology and/or serve as a new memory tier. When a fixed programming voltage is applied to the resistive cell (memristor), its resistance changes logarithmically in time. This is undesirable for using the memristors as a multi-level cell (MLC) memory. We present Adaptive Programming (AP) – a feedback-based programming circuit and method that improve process variation tolerance and uniformity of MLC levels, by effectively linearizing the memristive behavior. An appropriate sneak-path current mitigation has been identified as well. AP requires fewer programming steps than other programming methods, resulting in 46% faster programming and 95% energy reduction. AP reduces the frequency of errors (FoE) by 50% as compared to other writing schemes. Furthermore, AP enables using the memristors as a multi-level counter facilitating multi-valued computing arrays for non-von Neumann machines.

Keywords: MLC, storage, flash, ReRAM, memristor, memristive systems, non-von Neumann

1. Introduction

Resistive Random-Access Memories (ReRAMs) are memory technologies where data is stored within a resistive switch element (namely, a memristor) as the resistance of the device. Storing data as resistance rather than electric charge (as in DRAM) allows longer data retention. Since no charge is stored within the cell, there is no leakage. Numerous materials have shown memristive behavior (*i.e.*, varying resistance). These materials include many oxides and other dielectric materials, which can therefore be used to design ReRAM in the back-end-of-the-line (BEOL) CMOS process, as a crosspoint between metals. ReRAM therefore has the potential to be extremely dense, low power, with high endurance, making it an attractive technology for secondary storage and storage class memory tiers. The change in memristor resistance is continuous and can therefore be programmed to different values [1–4], which allows the design of multi-level memory cells (*i.e.*, cells that store more than a single bit) to further increase memory capacity.

In nanoscale memristive devices, high electrical fields and elevated temperatures lead to significant nonlinearities in ionic transport [5]. These nonlinearities are mostly observed at the thin film edges of a memristor [6–8]. Therefore, considering the nonlinear nature of memristor dynamic behavior, resistance levels of the device cannot be uniformly distributed while using identical fixed programming voltage pulses to program a MLC, namely, while using identical pulse programming (IPP). Thus, the transition between different resistance levels requires several different programming steps with different voltage pulses for each step [9]. Various ReRAM programming techniques, some of them presented in sub-section 3.2, have been reported in the literature, including staircase programming [4], [10], program and verify [4], [11], self-controlled programming [12], and time-domain state control [13]. These techniques suffer from increased latency, energy, area and complexity. Furthermore, potential comparator-based memristor state tuning methods [14] might dramatically increase area and energy as well if they were adopted in ReRAM.

Using constant voltage pulses to program the memristive devices while uniformly distributing the resistance levels of MLC ReRAM would improve process variation tolerance and decrease programming steps. Hence, constant pulse programming would reduce latency and complexity as compared to other programming tech-

*These authors share an equal contribution

Email addresses: misbah.ramadan@gmail.com (Misbah Ramadan), nicolasw@campus.technion.ac.il (Nicolás Wainstein), ran@ee.technion.ac.il (Ran Ginosar), shahar@ee.technion.ac.il (Shahar Kvatinsky)

[1]Corresponding Author

niques. In this paper, we present Adaptive Programming (AP), where only fixed voltage pulses are used to program the memristor to different resistance values, while effectively linearizing the behavior of the memory cell using feedback peripheral circuits. The feedback mechanism transforms the fixed voltage pulse into the required applied voltage according to the current and the desired resistance levels in a simple and elegant manner. AP improves the speed of programming and its process variation tolerance by achieving uniformity in the distribution of memristor resistance as compared to previously proposed identical pulse programming (IPP) methods [15]. While AP simplifies the write mechanism for memory applications, this technique can also be used as a building block for in-memory multi-valued computing [16]. This novel writing scheme is evaluated using HfO_2 memristors, which are used as memory cells in memristive crossbars.

The rest of the paper is organized as follows. Memristive technologies are introduced in Section 2. Related work and motivation behind uniformly distributing resistance levels and its impact on process variation tolerance and latency are described in Section 3. In Section 4, the Adaptive Programming method is presented. The applicability of AP within a resistive crossbar memory array is demonstrated in Section 5. Evaluation of adaptive programming and comparison to other programming techniques is presented in Section 6 and in Section 7 different approaches to overcoming operational amplifier offset-voltage are discussed. In-memory computing using AP and related challenges are discussed in Section 8, followed by concluding remarks in Section 9.

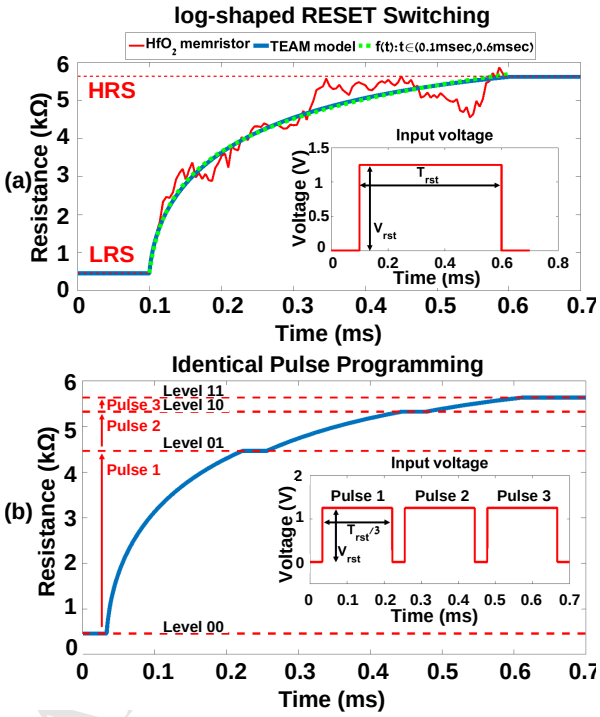

Figure 1: (a) RESET switching of tuned TEAM model (blue line) [20] that mimics the physical behavior of an HfO_2 memristor [21] (red line) under CVS. The tuned parameters are listed in Table 1. The TEAM model is also fitted by $f(t) = 1325.1 \cdot ln(10^7 t - 900) - 5627.7$ for $t \in (0.1\ ms,\ 0.6\ ms)$. (b) Resistance levels are distributed in a non-uniform manner when using identical pulse programming with $3V_{rst}$ pulses of length $T_{rst}/3$.

2. Memristive Technologies

Memristive devices are non-volatile programmable resistors that can be fabricated in intersection point between rows and columns of a metal grid (*i.e.*, crossbar). Nowadays, there are different memristive technologies and among the most mature of them are, Phase-Change Memory (PCM) [17], Spin-Transfer Torque Magnetic RAM (STT-MRAM) [18], and Resistive RAM [19]. The ReRAM structure is composed of resistance-changeable materials, usually metal-oxides such as TiO_2, HfO_2 [19], sandwiched between two-terminal metal electrodes. The change of the resistance can be achieved by passing current or applying voltage on these electrodes, which can lead to the formation or rupture of conductive laments (metallic or decomposed sub-oxide) between the electrodes that decreases or increases the resistance, respectively. The switching in most ReRAM devices is bipolar, *i.e.*, by applying positive or negative voltages across the cell terminals the cell switches from low-resistive state (LRS, R_{ON}) to high-resistive state (HRS, R_{OFF}), or from HRS to LRS, respectively, while possibly achieving HRS/LRS ratio >10 [19]. Since HfO_2 is used as a high-k dielectric for the gate insulator in MOSFET, the device can be easily integrated to the CMOS fabrication process. Furthermore, since the fabrication temperature is BEOL compatible, ReRAM can be stacked in a three dimensional fashion with an effective memory area of $4F^2/n$, where n is the number of stacked layers [19].

Compared to other memristive technologies, ReRAM presents a better endurance than PCM ($>10^9$) and lower writing latencies [22]. STT-MRAM exhibits better endurance but has lower R_{OFF}/R_{ON} ratios, and it is not possible to program multiple levels in one cell. Hence, ReRAM technologies are attractive candidates to replace current FLASH technologies, and furtherly extend their features. ReRAM technologies can achieve high-performance operations, in the order of nanoseconds for both read and write operations, compared to

hundreds nanoseconds for reads and microseconds for writes in FLASH [23], while retaining relatively low programming voltages as compared to FLASH (~12-22 V [24]). Also ReRAM can substantially extend FLASH endurance from approximately 10^3 erase operations [25] to approximately 10^8 program cycles [26], while featuring non-volatility and multi-level capability [1] by applying programming pulses to control intermediate resistant levels and maintaining the memory structure.

3. The Need for an Accurate Resistive-Level Control Mechanism

Different programming methods are used to accurately program the memory cells, while dealing with the impact of process variations on resistance level distribution in MLC. Studies of the effect of programming operations on the resistance of the cell show that the switching of memristors is nonlinear in a logarithmic manner [6–8], as depicted in Fig. 1(a) for a RESET operation of an experimental bipolar HfO_2 memristive device, reported to have a gradual RESET behavior but an abrupt SET behavior.

Cells are programmed in ReRAM by applying a constant voltage stress (CVS) [27] across the cell, for a sufficient amount of time. In Single Level Cell (SLC), two programming operations can be performed, RESET and SET, which program the cell, respectively, from a low resistance state (LRS) to a high resistance state (HRS) and vice versa. Both RESET, and SET are performed using a CVS of V_{rst} and V_{set} for a sufficient length of time, namely, T_{rst} and T_{set}. Alternatively, for MLC ReRAM, the levels are represented by different resistances of the cell in addition to the two boundary resistance levels, LRS and HRS. Hence, programming operations in MLC rely on using finer voltage pulses that achieve multiple resistance levels. Due to the logarithmic transition of the memristive device, applying identical voltage pulses changes the resistance in a non-uniform manner across the resistance range, as shown in Fig. 1(b). Closely located levels might result in increased sensitivity to process variations and erroneous reading of cells. Section 3.1 discusses the effect of non-linearity on process variation tolerance of the MLC ReRAM. Previously proposed programming methods and their effect on programming latency and energy are discussed in Section 3.2.

Table 1: Parameters of the TEAM Model

	Physical Device TiN-HfO_2-Pt			
	OFF Region (RESET)		**ON Region (SET)**	
Parameters of TEAM model [20]	R_{off}	5.63 kΩ	R_{on}	460 Ω
	k_{off}	$0.0035\ m/s$	k_{on}	-10 m/s
	α_{off}	1	α_{on}	1
	i_{off}	0.202 mA	i_{on}	-0.14
	x_{off}	0 m	$x_{on} = D$	10^{-6} m

3.1. Impact of Non-linearity and Process Variations on Level Distribution

Process variation in memristors results in resistance variation of the cells across the memory array. Consider an N-level cell enabling N different levels of resistance, thereby storing N different values (equivalently, k bits, $2^k = N$). Due to process variation, each resistance level is distributed over a certain range, as illustrated in Fig. 2(a). The range of level i is characterized by the slowest and fastest cells in the entire ReRAM array, R_i^S and R_i^F. A reference resistance value $R_{ref}^{i,i+1}$ is used to differentiate between distributions of level i and level $i+1$. Hence, a read operation determines the resistance level of a cell by performing multiple comparisons. For example, a cell is at level i if its resistance (*i.e.*, R_{cell}) fulfills the following:

$$R_{ref}^{i-1,i} \leq R_{cell} \leq R_{ref}^{i,i+1}. \quad (1)$$

In N-level ReRAM, $N-1$ reference resistors are required to differentiate between the N levels. Nevertheless, since each level is defined by a resistance range, some level ranges might overlap when using identical pulses to program a cell, namely,

$$\exists\, i, R_i^S < R_{i-1}^F, \quad (2)$$

which may lead to an erroneous read of the overlapping cells, as illustrated in Fig. 2(b). Linearizing the memristor time-to-resistance programming function while taking process variations into account can result in uniformly distributed levels across the resistance range, thus reducing the overlapping regions, as shown in Fig. 2(c). Linearization of that function can therefore increase the capacity of an MLC (*i.e.*, the number of stored bits per cell) and reduce the redundancy of the ReRAM by using fewer bits for error correcting codes (ECC) [28], required for reliable delivery of the digital data.

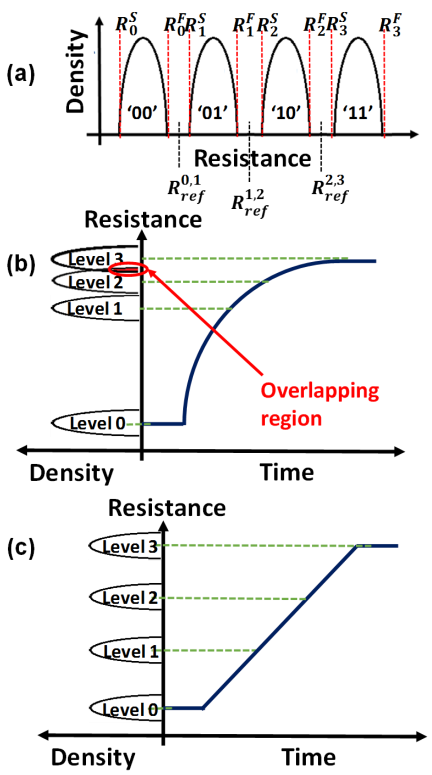

Figure 2: (a) Each level i range is defined by the fastest and slowest cells, respectively, R_i^F and R_i^S. Whether the resistance transition of a memristor is (b) nonlinear, or (c) linear, determines whether the levels are distributed non-uniformly or uniformly when using identical pulses for programming.

3.2. *Programming Techniques and Their Effect on Latency, Energy, and Complexity*

Different programming techniques have been proposed to overcome the impact of non-linearity on MLC level distribution and increase the tolerance to process variation. Two intuitive techniques to compensate for the slowdown in the resistance transition rate under CVS are to increase the applied voltage magnitude (*Incremental Magnitude Pulse Programming*, IMPP, Fig. 3(a)) and to increase the duration (*Incremental Length Pulse Programming*, ILPP, Fig. 3(b)) [15]. Both ILPP and IMPP require a read operation prior to programming to adjust the applied pulses to the current and desired states of the memristor. Hence, both are state-dependent pulsing techniques. The extra read operation increases programming latency, complicates the voltage generators, and requires a controller. Furthermore, state-dependency restricts in-memory multi-valued computing, as further discussed in Section 6.

Another programming method to control uniform

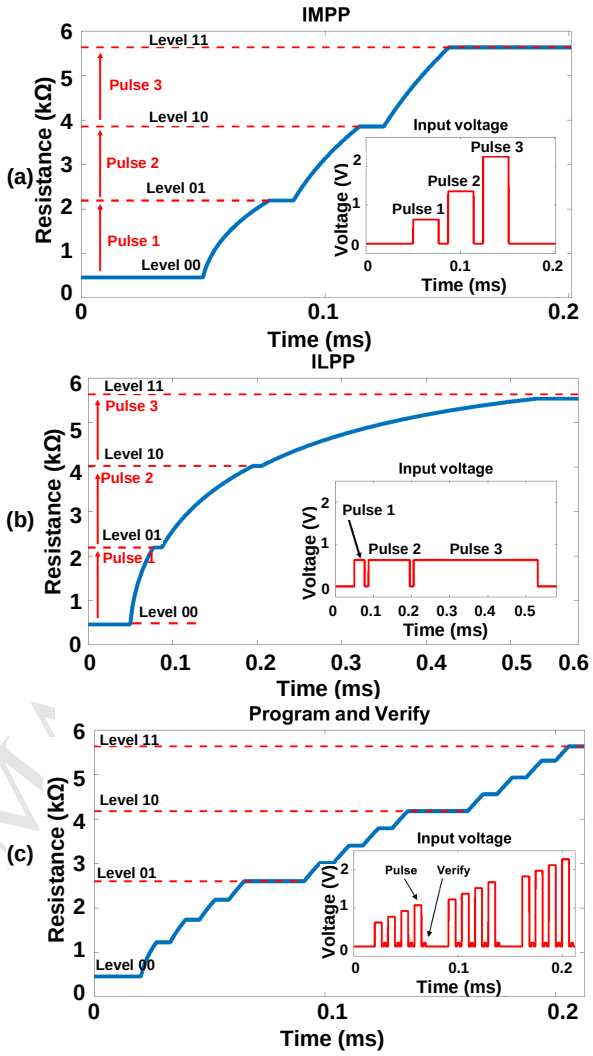

Figure 3: Programming multi-level memristor cell using (a) IMPP, (b) ILPP, and (c) P&V

level distribution is *Program and Verify* (P&V) [3, 11, 27]. The P&V method relies on applying narrow voltage pulses with progressively increasing magnitude, combined with read pulses between each programming pulse to verify whether the cell has reached the desired state, as illustrated in Fig. 3(c). Thus, P&V takes variations into account. In ReRAM, programming operations are usually initiated on a word line [4], and the controller is responsible for connecting or disconnecting the cells when the desired state is achieved. Hence, P&V allows selective programming of cells while controlling tighter resistance level distributions, possibly allowing higher capacity. However, P&V requires more programming iterations as compared to ILPP and IMPP,

and enhanced interference from the controller, which increases latency of the programming operation. Hence, it would be beneficial to develop a programming technique with fewer programming iterations and voltage generators (ideally, a single applied voltage) for less energy and lower latency. To achieve uniform level distribution by applying identical pulses that correspond to the desired level, the memristor should respond linearly under CVS, as in Fig. 2(c). For example, for an N-level cell with linear transition under CVS of length T, a level i can be programmed using i identical pulses of length $T/(N-1)$. As shown in the next section, Adaptive Programming enables such a linear response.

In [12], a bank of resistors is used to control the voltage of the device to achieve a defined level. The cell forms a voltage divider with the bank of resistors, thus adapting the voltage drop on the cell according to the resistor value. Unlike most of the MLC programming methods, the *Self-Controlled Multilevel Writing* (SCMLW) intends to control the abrupt SET operation. Since the maximum voltage is constrained by the voltage divider, a constant width and constant amplitude pulse is enough to guarantee the proper voltage drop on the device. This method suffers from large area overhead as large resistor values are required (≥10 kΩ). For instance, a 1 MΩ polysilicon resistor in the 180 nm technology node has an approximate length of 500 μm. Thus, an increment in the number of levels, implies additional area overhead. Furthermore, the well-known the large process and cycle-to-cycle variations of the resistors from flash analog-to-digital converters undermines the reliability of this approach.

4. Adaptive Programming

Adaptive Programming (AP) adapts identical voltage pulses into variable pulses to induce a linear time-to-resistance programming response. AP is illustrated in Fig. 4 and operates as follows. Depending on the present and desired resistance levels, a controller applies k pulses of constant voltage. The resulting resistance level is continuously monitored by the feedback circuit and an appropriate compensation voltage is added to the input constant voltage pulses. At the end of the k-pulse sequence, the desired linearly-mapped resistance level is achieved. The AP circuit comprises an inverting operational amplifier with the memristors in the negative feedback, as shown in Fig. 4(a). Assume an ideal operational amplifier. Then, the voltage across the memristors is

$$V_{mem}(t) = V_{out}(t) = -V_{in}\frac{R_{mem}(t)}{R_{in}}, \qquad (3)$$

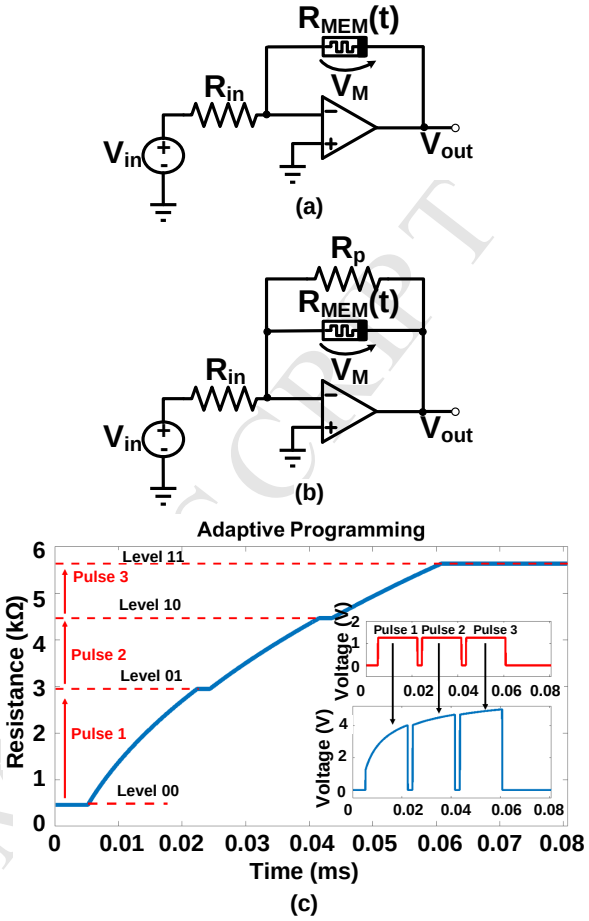

Figure 4: Adaptive programming consists of a (a) memristor connected in the negative feedback of an operational amplifier in inverter configuration. (b) To avoid saturation of the amplifier, the proposed Adaptive Programming technique consists of an additional resistor R_p. (c) Using AP allows identical pulses to be amplified adaptively, leading to a more linear-like transition and better level distribution as compared to identical pulses programming.

where V_{in} is an input voltage that consists of identical voltage pulses. During programming, the voltage across the memristor is greater than the device threshold [15], *i.e.*,

$$|V_{out}(t)| = \left|V_{in}\frac{R_{mem}(t)}{R_{in}}\right| > |V_{th}|. \qquad (4)$$

In practice, the circuit shown in Fig. 4(b) is employed. To ensure (4), V_{in} may need to be increased to a level that might saturate the operational amplifier. The bounding case is when $R_{mem}(t)$ is at its minimal value, LRS. With V_{in} sufficiently high given $R_{mem}(t) = LRS$ (*i.e.*, the resulting V_{out} is greater than V_{th}), then, when $R_{mem}(t) = HRS$, the resulting V_{out} may exceed the saturation voltage of the operational amplifier. To mitigate that issue, an additional resistor R_p is connected in par-

allel with the memristors. The resistances of R_{in} and R_p are selected to achieve voltage compensation across the entire LRS to HRS range as follows.

$$\left| V_{in} \frac{LRS \| R_p}{R_{in}} \right| > |V_{th}|, \tag{5}$$

$$\left| V_{in} \frac{HRS \| R_p}{R_{in}} \right| \leq |V_{DD}|. \tag{6}$$

To achieve a full transition across the entire resistance range, R_{in} and R_p are set after solving the pair of equations (5) and (6) in their equality form, given V_{in}, LRS, HRS, V_{DD}, and V_{th} as constants, and the pair R_{in} and R_p as variables. After selecting R_{in} and R_p, a linear-like transition is achieved under CVS as depicted in Fig. 4(c).

The proposed circuit for adaptive programming allows the controller to apply identical V_{in} pulses at the input. The circuit adaptively amplifies them according to the state of the memristors. The adaptive amplification eliminates the state-pulse dependency between resistance levels and voltage pulses. Furthermore, AP with identical pulses distributes the resistance levels uniformly across the resistance range and improves the tolerance to process variation, while maintaining simplicity of the programming mechanism. Given the LRS-HRS range of memristor resistance, the solution selects appropriate parameters V_{in}, R_{in}, R_p and appropriate ranges of operation that lead to $R_{mem}(t)$ being approximately linearly time-dependent within the LRS-HRS range, with a bounded diversion from exact linear dependence. As seen in Fig. 4(c), the rate of change of the memristor resistance is higher than that of $V_{out}(t)$. This behavior is due to the damping effect of R_p in

$$V_{out}(t) = \frac{|V_{in}|}{R_{in}} \left(R_{mem}(t) \parallel R_p \right). \tag{7}$$

5. Crossbar Compatibility

The design of Adaptive Programming using operational amplifiers is compatible with the crossbar array, the basic topology underlying ReRAM, and can be feasibly embedded within the ReRAM peripheral circuitry.

In crossbar architecture, each memory cell (*i.e.*, memristor) is placed in column-row intersection as depicted in Fig. 5. A cell can be accessed either for reading or programming by accessing the corresponding column and row of the cell within the crossbar. The cell located in row i and column j is read by applying a read voltage, lower than the memristor threshold, on row i and sensing the current flowing through column j. Write operations are performed by applying a program volt-

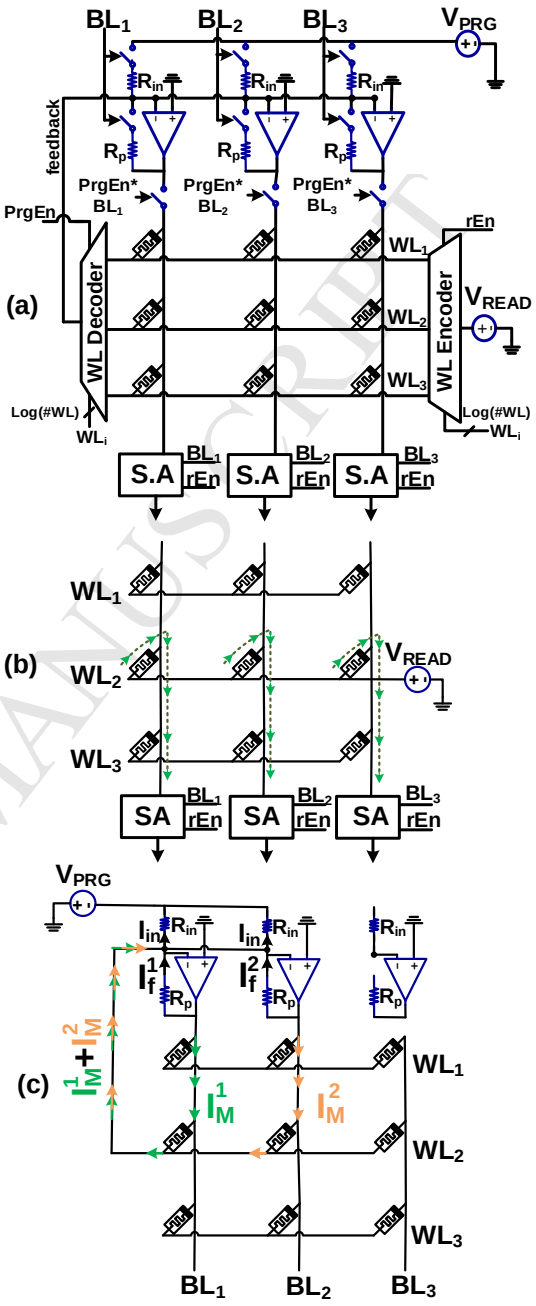

Figure 5: 3×3 ReRAM crossbar array supporting adaptive programming. (b) Functional circuitry when performing parallel read operation of WL_2. In this operation, rEn is set and the right WL decoder chooses WL_2. (c) Functional circuitry when performing parallel program operation of BLs 1 and 2 within WL_2 using adaptive programming. In this operation, $PrgEn$ and both BL_1 and BL_2 are set and the left WL decoder chooses WL_2.

age on column j and connecting row i to the ground. To change the device resistance, the programming voltage should be greater than the memristor voltage threshold. In bipolar memristors, the voltage polarity (negative or positive) indicates whether the resistance will increase or decrease.

To enhance ReRAM performance, operations can be performed simultaneously on the entire row (wordline, WL). An entire wordline is read by applying a read voltage on it and sensing the currents in all columns (bitlines, BLs). Programming operations, SET and RESET are performed separately in WL granularity, by applying a program voltage to all bitlines within a selected WL, and connecting the corresponding WL to the ground. The opamps needed to enable the AP method are included one per column, rather than per cell (Fig. 5). The read circuitry is similar to conventional ReRAM array topology and supports reading of an entire WL (Fig. 5(b)). Writing (programming), however, employs the operational amplifiers (Fig. 5(c)), as follows. First, SET is applied to an entire row. Next, a sequence of RESET operations is performed over the entire row, selectively to cells enabled by their respective BLs. Note that all BLs share the same feedback connection; the number of activated BLs is the same as the number of connected R_{in} resistors (Fig. 5(a)).

6. Design and Evaluation

In this section, the operational-amplifier-based adaptive programming (AP) circuit is presented and evaluated in simulations, and our method is compared to previously proposed circuit programming techniques. Two types of simulations are used, SPICE circuit simulation and MATLAB Monte Carlo simulation for a 3×3 crossbar.

6.1. Circuit Simulations

A ReRAM circuit based on a 180 nm CMOS process is designed in Cadence Virtuoso. The opamp and the sense amplifier are modeled using the parameters listed in Table 2. The TEAM model [20] is fitted [29] to the switching dynamics of an HfO_2-based bipolar memristor, which reportedly has a gradual RESET operation while exhibiting a sub-nanosecond SET operation. TEAM, with its fitted parameters listed in Table 1, models the logarithmic response of that memristor to CVS as demonstrated in Fig. 1. Since the parasitic capacitance of the memristive device was not characterized, it is approximated to that of a parallel plate capacitor for a $4F^2$ device in 180 nm process leading to $C_P = 1.15$ fF. The

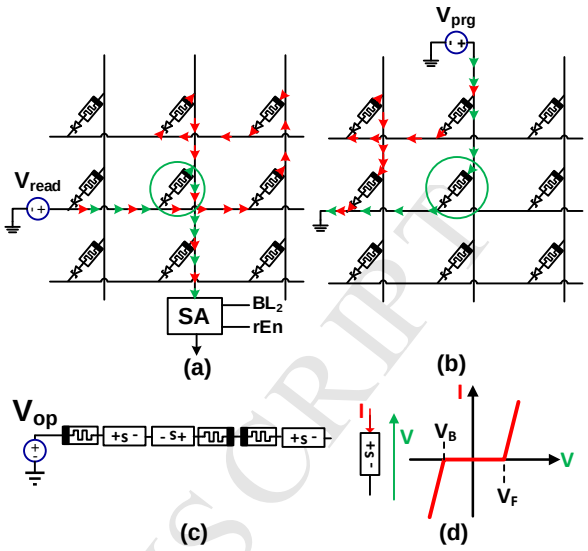

Figure 6: Sneak path (red arrows) during (a) a read operation of a cell (green circle), and during (b) a program operation of a cell (green circle). (c) Sneak path of length 3 in a 1S1R crossbar, (d) First order I-V characteristics of a selector.

switches were implemented using a wide PMOS transistor ($W = 10\ \mu$m). Since the timing is not critical, in this work a slow rise time was used to remove issues related to clock feedthrough. Charge injection noise was not considered in this work and is left as future work. These issues can be solved using a bootstrapping transistor and bottom plate sampling [30].

In crossbar arrays, current can sneak into different paths other than the desired paths for read and program [31, 32]. These sneak path currents can cause write disturbances (changing cell states undesirably along the sneak path) or cause an erroneous read operation due to the increased currents sensed by the sense amplifiers, as illustrated in Fig. 6. Several solutions have been proposed to solve the sneak path problem, such as unfolded crossbar architecture [31] , diode gating (1D1M) [31], transistor gating (1T1R, 2T1R, 4T1R) [31], selectors [33, 34], complimentary resistive switching (CRS) [35, 36] , half-selected cells [1], and coding [37]. However, if we consider the different programming schemes used to program MLC ReRAM by increasing magnitude voltage pulses, some of the proposed methods for sneak-path mitigation will not work as effectively for MLC ReRAM as they do for SLC ReRAM. For example, half-selecting cells [1] need to adapt to the change in programming voltage to keep other cells half-selected and not cause write disturbances. Providing variable voltage requires complex voltage regulators and peripheral circuitry. CRS is only

Table 2: Simulation Parameters for AP

Parameter	Symbol	Value
Read voltage	V_{READ}	-0.6 V
Operational Amplifier		
Supply voltage	V_{CC}	5 V
Gain	G	60 dB
Gain-bandwidth product	GBW	10 MHz
Intrinsic input resistance	R_{input}	1 MΩ
Slew Rate	SR	0.5 V/μs
Output Resistance	R_{out}	80 Ω
In resistance	R_{in}	332 Ω
Feedback resistance	R_P	1.687 Ω
Sense Amplifier		
Gain	A_u	60 dB
Settling time	τ	0.2 μs

compatible with SLC ReRAM, while unfolded architecture increases power, latency, and area overhead [38]. Nevertheless, transistor gating methods increase area dramatically, and thus, might be less suitable to adopt AP programming methods. Furthermore, coding requires extra bits for its implementation, increasing the footprint of the memory array [37].

We consider an asymmetrical selector [39] (see I-V characteristics in Fig. 6(d)) to mitigate sneak paths during program and read operations with minimal density and fabrication overhead. The selector exhibits a forward voltage threshold of V_F, and a backward voltage threshold of V_B. Therefore, considering the worst-case sneak path of length three in the 3×3 crossbar (see Fig. 6(c)), the operational voltage for reading should fulfill the following constraint,

$$V_F < V_{read} < 2V_F + |V_B|, \tag{8}$$

meaning sufficiently high to make one selector conductive, but low enough not to open the remaining selectors. For programming the operational voltage should allow current flow in the desired path for operation, while suppressing sneak path currents. These constraints are

$$V_F + V_{th} \leq V_{op} \leq 2V_F + |V_B| + \frac{R_{off}}{R_{off} + 2R_{on}} V_{th}, V_{op} > 0, \tag{9}$$

$$|V_B| + V_{th} \leq \left|V_{op}\right| < 2|V_B| + V_F + \frac{R_{off}}{R_{off} + 2R_{on}} V_{th}, V_{op} < 0, \tag{10}$$

where V_{op} is the operational voltage for either reading or programming a cell, and V_{th} is the threshold voltage of the memristor.

However, for selectors with symmetric characteristics, the constraints become

$$V_S + V_{th} \leq V_{op} \leq 3V_S + \frac{R_{off}}{R_{off} + 2R_{on}} V_{th}, V_{op} > 0, \tag{11}$$

$$V_S + V_{th} \leq \left|V_{op}\right| < 3V_S + \frac{R_{off}}{R_{off} + 2R_{on}} V_{th}, V_{op} < 0, \tag{12}$$

where $\pm V_S$ is the breakthrough voltage of the selector. Hence, with symmetrical selectors, and considering $R_{off} >> R_{on}$, the required operational voltage range is $|V_{op}^{max}| - |V_{op}^{min}| = 2V_S$ to ensure correct read and program operations. The operational voltages when using asymmetrical selectors are, however, restricted to the range of $|V_{op}^{max}| - |V_{op}^{min}| = V_F + |V_B|$, when $|V_B| >> |V_S|$. Asymmetrical selectors enable a wider range of operational voltages for programming operations as compared to symmetrical selectors. Therefore, we have chosen to use asymmetrical selectors in 1S1R topology for their feasibility with the different programming methods and their compatibility with high programming voltages. The SPICE model for the asymmetrical selector we used in our simulations can be found in the appendix. The selector is modeled as a Zener diode in series with a regular diode, the parameters were fitted to match those in [39].

Simulation results of AP in a 3×3 memory array for a whole WL are shown in Fig. 7. Prior to programming, a SET operation is applied to the entire WL. Subsequently, identical pulses at the input are applied and are adaptively amplified. The number of identical pulses is determined by the controller according to the desired level, and the BL of each cell is disconnected when reaching the desired number of pulses (as per Fig. 5).

We compare different programming methods (cf. Section 3) by SPICE simulations, optimizing the voltage pulse magnitude and length of each method, as listed in Tables 2 and 3. For IPP, voltage pulses are chosen to achieve the optimal programming latency at the expense of programming energy, while in ILPP and IMPP, the voltage pulse length and magnitude (within the operational voltage range $V_{op} \in [-5V, 5V]$) are selected to achieve the best level distribution of a four-level cell resistance.

In the P&V method, the voltage pulse amplitude or duration trades off the controlled resolution of the resistance levels. Higher or wider voltage pulses increase the resistance steps more rapidly and as a result lower the resolution control. The voltage pulses therefore have been chosen to achieve sufficient accuracy with the aforementioned trade-off. In AP, R_{in} and R_p resistors are selected to achieve an appropriate voltage pulse amplification.

Table 3: Simulation Parameters for Different Programming Methods

Programming Method	Programming Voltage Pulses Needed for Each Level			
	Level 0 (SET)	Level 1	Level 2	Level 3
IPP	In all methods SET voltage and duration are $V_{SET} = -5\ V$ $T^0_{pulse} = 8.49\ ns$	$V^1_{pulse} = 4.9\ V$ $T^1_{pulse} = 1.7\ \mu s$	$V^2_{pulse} = 4.9\ V$ $T^2_{pulse} = 1.7\ \mu s$	$V^3_{pulse} = 4.9\ V$ $T^3_{pulse} = 1.7\ \mu s$
IMPP		$V^1_{pulse} = 1.75\ V$ $T^1_{pulse} = 2.65\ \mu s$	$V^2_{pulse} = 3.4\ V$ $T^2_{pulse} = 2.65\ \mu s$	$V^3_{pulse} = 4.9\ V$ $T^2_{pulse} = 2.65\ \mu s$
ILPP		$V^1_{pulse} = 1.75\ V$ $T^1_{pulse} = 2.65\ \mu s$	$V^2_{pulse} = 1.75\ V$ $T^2_{pulse} = 10\ \mu s$	$V^3_{pulse} = 1.75\ V$ $T^3_{pulse} = 30\ \mu s$
P&V		Each programming pulse is increased by 0.2 V starting from $V_{init} = 1.75\ V$. Each verify / read pulse is $V_{read} = -0.6\ V$. All pulses are of length $T_{pulse} = 0.5\mu s$. The number of pulses used to program a cell to level 1, 2, and 3, where 5, 11, and 17, respectively.		
AP		To program a cell to level i, i identical pulses are applied $V_{pulse} = 1.25\ V$ $T_{pulse} = 1.9\ \mu s$.		

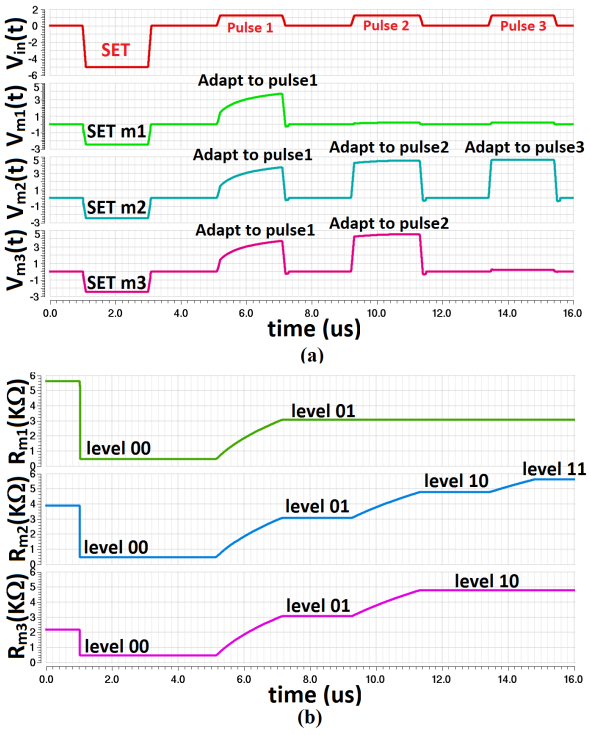

Figure 7: Programming WL_2 using AP. The pattern '01', '11', '10' is written, respectively, to the cells at BL_1, BL_2, and BL_3. (a) First a SET operation is performed to SET all cells within WL_2 and then identical pulses are applied (V_{in}), which are adaptively amplified according to the cell's resistance (V_{m_i} describes the voltage dropping on the memristor at BL_i). (b) The resistance of memristor R_{m_i} (*i.e.*, the memristor at BL_i).

Energy and latency (per cell) results for different programming methods are listed in Table 4. Adaptive programming achieves the lowest latency as compared to other programming methods except IPP, improving programming speed by 29% as compared to IMPP. For IPP, voltage pulses have been chosen to achieve the lowest programming latency possible (*i.e.*, pulses have been chosen with high magnitudes) at the cost of programming energy; however, IPP still suffers from reliability issues much more than AP, as discussed in the following sub-section.

Note that the P&V method also causes energy dissipation during the verify pulse (*i.e.*, read pulse) after each programming pulse. Since many optimizations can be applied to the read circuitry, we took the best known case of read energy and latency (1.41 nJ and 1.61 ns in a 22 nm CMOS process [11]) and multiplied those values by the number of verify pulses required for the programmed cells to reach levels 1, 2, and 3 (×5, 11, and 17, respectively). The energy and latency figures are extrapolated for a 180 nm CMOS process according to the scaling estimates of [40], as listed in Table 4. Simulations show that AP reduces energy dissipation up to 95% and reduces programming latency by at least 46% as compared to P&V.

6.2. Statistical Analysis

We performed MATLAB based statistical analysis to evaluate the improvement of the AP method under process variations while considering Frequency of Error (FoE) among ReRAM cells as a figure of merit. We evaluated the impact of process variation on the level

Table 4: Energy and Latency Comparison of the Different Programming Methods

Programming Method	Level 0 (SET)[a]		Level 1[b]		Level 2[b]		Level 3[b]	
	Energy (fJ)	Latency (ns)	Energy (nJ)	Latency (μs)	Energy (nJ)	Latency (μs)	Energy (nJ)	Latency (μs)
IPP	106.6	8.5	17.89	1.7	26.72	2.4	33.67	3.1
IMPP			4.2	2.7	12.7	5.3	24.4	7.9
ILPP			5.9	2.7	14.2	12.7	25	42.7
P&V			146.2[c]	3.51	348.8[c]	7.7[c]	516[c]	11.9[c]
AP			10.6	1.9	19.6	3.8	27.2	5.7

[a] Energy and latency of SET operations are calculated according to the worst case, which is setting the device from level 3 to level 0.

[b] Energy and latency calculated after SET operation.

[c] Values including reading (verify) energy and latency reported in [10] (for a 22nm CMOS process) and extrapolated to a 180 nm process according to [26].

distribution of MLC in different programming techniques. First, the impact of process variations on memristors using the TEAM model was studied. To add process variations, we adopted the methods used to determine the influence of variations in the linear ion drift memristor model [6] from [41, 42]. The TEAM model [20] consists of the two following expressions,

$$\frac{dx}{dt} = \begin{cases} k_{on}\left(\frac{i(t)}{i_{on}} - 1\right)^{\alpha_{on}} f_{on}(x), & i < i_{on} < 0, \\ 0, & i_{on} < i < i_{off}, \\ k_{off}\left(\frac{i(t)}{i_{off}} - 1\right)^{\alpha_{off}} f_{off}(x), & 0 < i_{off} < i, \end{cases} \tag{13}$$

$$R(t) = R_{on} + \frac{R_{off} - R_{on}}{D} x, \tag{14}$$

where $x \in [0, D]$ is an internal state variable, D, k_{off}, α_{off}, and α_{on} are positive constants, k_{on} is a negative constant, i_{off} and i_{on} are current thresholds, and R_{on} and R_{off} are, respectively, the LRS and HRS. The current and resistance of the device are, respectively, $i(t)$ and $R(t)$. The resistance of the TEAM model can exhibit, among other I-V relationships, a linear I-V relationship as described by (14), similar to the I-V relationship in the linear ion drift model. Therefore, the same process variation models in [41, 42] for modeling parameter distribution under process variations can be adopted for the TEAM model. In this model, the parameters R_{off}, R_{on}, and D vary in a normal distribution,

$$R_{off} \sim N\left(\mu_{R_{off}}, \sigma^2_{R_{off}}\right), \tag{15}$$

$$R_{on} \sim N\left(\mu_{R_{on}}, \sigma^2_{R_{on}}\right), \tag{16}$$

$$D \sim N\left(\mu_D, \sigma^2_D\right), \tag{17}$$

where $\mu_{R_{off}}$, $\mu_{R_{on}}$, and μ_D are, respectively, the desired values of R_{off}, R_{on}, and D without process variations (*i.e.*, 5.63 kΩ, 460 Ω, and 10^{-6} m). Their values are $\sigma_{R_{off}}$, $\sigma_{R_{on}}$, and σ_D are the standard deviations of, respectively, R_{off}, R_{on}, and D.

To perform process-variation aware simulations, we built a MATLAB-based Monte Carlo (MC) simulation environment. As a figure of merit, we chose the Frequency of Error (FoE), which is the frequency of occurrence of different ReRAM chips suffering from the same number of bit errors, while an error refers to an erroneous read operation of a cell that was intended to be programmed to level i, but whose resistance is lower than R^F_{i-1} (*i.e.*, lower than the fastest cell of the next lower level) or higher than R^S_{i+1} (*i.e.*, greater than the slowest cell of the previous lower level). Other possible metrics are mean-time between failures (MTBF) and bit error rate (BER).

We compared AP to identical pulse programming (IPP) as they both have state-pulse independency and the same underlying programming mechanism, which is identical pulses. We sampled 10,000 MC samples and wrapped them in 100 iterations (*i.e.*, simulating 100 ReRAM chips) to induce the frequency of bit errors, while considering different deviations for the distributions and the possibility of increasing the cells' capacity. For process variation with standard deviation of 2%, adaptive programming completely eliminates the bit errors, as shown in Fig. 8(a). When process variations are more severe, the frequency of errors of adaptive programming is approximately 50% lower (better) than IPP, as shown in Fig. 8(b). Note that the other programming techniques (*i.e.*, IMPP, ILPP, and P&V) achieve better uniformity with process variations as compared to the proposed AP design, and their FoE is negligible in practice. Since AP outperforms these methods and has

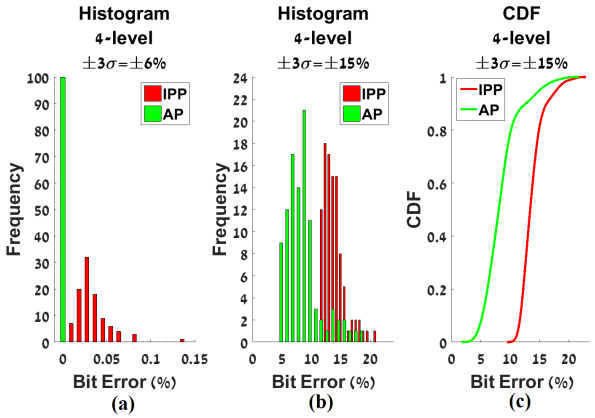

Figure 8: Frequency of bit error for IPP (red) and AP (green) with standard deviation of (a) $\sigma = 2\%$ and (b) $\sigma = 5\%$, and the (c) Cumulative Distribution Function of bit errors.

a simpler mechanism, we believe that an upgraded circuit design of AP with better linearity will also improve the uniformity, while maintaining the superior performance. Furthermore, IPP and AP enable in-memory computing since the identical pulse programming mechanism is an important block in some of those architectures, as discussed in the following section. Our results show that AP has a clear advantage over IPP in terms of performance, energy and reliability (FoE).

6.3. Area Analysis

An important figure of merit when discussing AP, is the area overhead of adding opamps to the peripheral circuitry. Adding any additional components to the peripheral circuitry will decrease the area efficiency of the memory module. The area efficiency is defined as the (array_area)(total_area), while the total_area includes the peripheral circuits. In the case of ReRAM, each memory cell can be fabricated in $4F^2$ area [43] (F is the feature size of the technology), which allows to build very dense memories. However, such memories can suffer from relatively low area efficiency due to the extreme dense memory cell fabrication in comparison to CMOS transistors. Therefore, the larger the memory crossbar is, the higher the area efficiency of the memory, since more cells share the same peripheral circuits.

To evaluate the area overhead of adding opamps, NVSim [43] is used. The tool allows dividing the memory into three main components, as depicted in Fig. 9(a). Each memory module is divided into banks that work separately, while each bank is composed of several memory mats that can work simultaneously. Each mat includes several resistive sub-arrays, that each one of them can grant access to a single cell (either for read or write). Therefore, to embed AP in such memories, an additional opamp should be added to the peripheral circuits of each sub-array, which will increase the peripheral area of the sub-arrays and the memory module in general.

A memory module including a single bank, comprised of 4×8 mats with 1×8 active mats at each cycle is simulated. Each mat includes 4×8 sub-arrays, while 1×8 sub-arrays work simultaneously at each cycle. For comparison, we considered a baseline circuit that exhibits a switch per column to control constant voltage pulses for programming, and we compared the area impact of adding opamps as part of AP on different array sizes from 32×32 to 1024×1024, while maintaining the same peripheral circuits such as sense amplifiers. Furthermore, the read and write latency of accessing the cells in the different sub-arrays and the impact of wire capacitance, resistance, and impact the different peripheral circuits on these parameters are measured. Also, the area impact of adding several opamps to the same array of size 512×512 is determined. The area and latency results are presented in Fig. 9(b) and Fig. 9(c), respectively, and the results of adding opamps shared among 2, 4, 8, 16, 32, and 64 columns are presented in Fig. 9(d). As can be noticed from Fig. 9(b), the area efficiency increases with sub-array size, and the reason for that is for small sub-array dimensions the dominant components of the total area are the peripheral circuit components. However, for read and write latency, the opamps almost has no effect (less than 1%) as can be seen in Fig. 9(c), but, the numbers increase with sub-array dimensions due to the increase in capacitance and resistance of the wires and the increased number of peripheral circuit components. As for adding several opamps in the same array, we can notice a more obvious impact on area, which decreases with the increase of the number of columns sharing the same opamp, as depicted in Fig. 9(d).

7. Overcoming Opamp Voltage Offset

The design of CMOS operational amplifiers (opamps) can be challenging when trying to achieve identical behavior between the different opamps due to mismatch between the transistors, and regularly results in variations in the offset voltage between these almost-identical opamps [44]. The offset voltage of an opamp is the voltage difference between the positive and negative inputs. In an ideal opamp, the offset voltage is zero, which means that the output voltage in open loop is zero. Denote by Δv_i the offset voltage of the opamp residing in BL_i and shown in Fig. 6(a). All

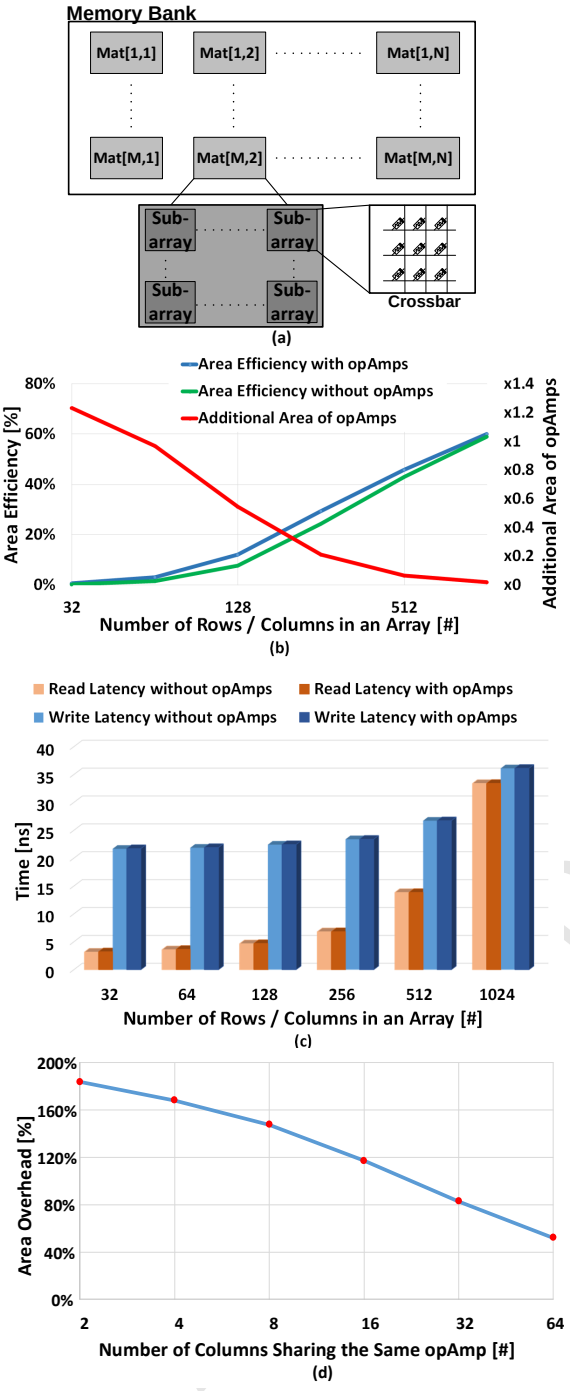

Figure 9: (a) Memory micro-architecture. The memory is divided to banks, mats, and sub-arrays. (b) Area simulation results comparing memory arrays with and without opamps, including area efficiency and additional area of opamps in percentages. (c) Comparing read and write latency of memory arrays with and without opamps. (d) Area overhead of adding opamps shared among several columns.

opamps in the proposed circuit share the same positive input, as well as the negative input,

$$v_{i,+} = 0 \; and \; v_{i,-} = v_- \rightarrow 0 \forall i, \tag{18}$$

where $v_{i,+}$ and $v_{i,-}$ are, respectively, the voltage of the positive and negative inputs of opamp i, and v_- is the common voltage value of all negative inputs. On the other hand, the output voltage of an ideal opamp can be expressed by

$$V_{i,out}(t) = A_i \left(v_{i,+} - v_{i,-}\right), \tag{19}$$

where A_i is the open-loop gain of the i-th opamp, and $V_{i,out}(t)$ is the output voltage (also the voltage across the memristor in AP) of opamp i. In ideal opamps, the open-loop gain A_i is sufficiently large to drive the offset voltage to zero for output voltages smaller than the saturation voltage driving the opamps, *i.e.*,

$$V_{i,out}(t) = \underbrace{A_i}_{\rightarrow\infty} \underbrace{\left(v_{i,+} - v_{i,-}\right)}_{\rightarrow 0} < V_{cc}. \tag{20}$$

Therefore, each opamp i tries to converge to the equilibrium point where $v_{i,-} \rightarrow -\Delta v_i$. Nevertheless, due to the physical connection of the opamps in Fig. 6(a) all negative inputs will be kept at the same potential, hence, not allowing a common equilibrium point. This in turn causes at least one output voltage of the opamp to always be at the saturation value V_{cc}, preventing proper functionality of the circuit.

In this section, we propose two solutions for adaptive programming with voltage offset to linearize the nonlinear switching of memristors using identical programming pulses. The first solution is based on changing the crossbar circuit topology. The second solution is to use current sources rather than opamps for the peripheral circuitry.

7.1. Semi-Crossbar Architecture

One possible solution to the offset voltage variation problem is to change the crossbar circuit from having a common feedback connection for all opamps to a separate feedback connection for each opamp. However, to achieve such a connection, the crossbar architecture should be modified into a Semi-Crossbar architecture with separate WLs for each BL, as depicted in Fig. 10. In the Semi-Crossbar architecture, read and program operations can still be performed simultaneously on a single cell per BL. However, in this architecture, the number of WLs is higher since every BL has its own WLs. Hence, the number of supported WLs in the memory array is $\#WLs \times \#BLs$, which increases complexity

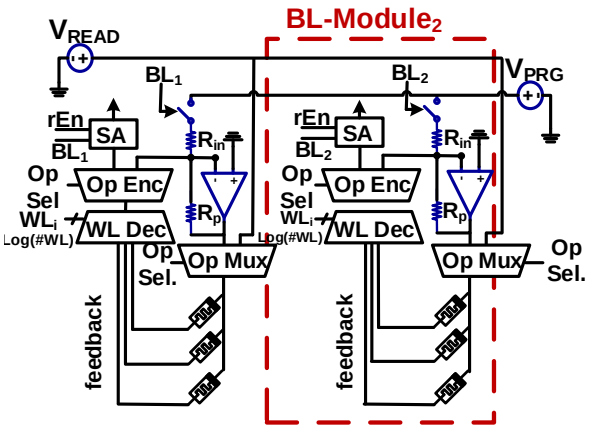

Figure 10: Semi-Crossbar architecture that exhibits separate WLs for each BL in order to allow separate feedback connections. Each BL-Module can be used for either programming or reading a single cell within the corresponding BL, and all BL-Modules operate simultaneously to allow parallel program or parallel read operations.

and limits the memory capacity. Although the capacity is lower, the semi-crossbar architecture completely eliminates sneak-path related issues. The semi-crossbar architecture is therefore more suitable for in-memory computing modules than large NVM, as discussed in Section 7.

7.2. Current Programming

Current programming techniques for ReRAM cells has been previously proposed [45]. Considering the proposed circuit for adaptive programming in Fig. 5(a) that translates identical voltage pulses to adaptively amplified voltage pulses, a similar behavior can be achieved using current programming circuits that use current pulses to program a cell consisting of a memristor and a selector in series, and a resistor connected in parallel to them both.

Assume ideal opamps, where each opamp has a virtual short, *i.e.*, $V_+ = V_- = 0$, and zero current flowing into either of its inputs. Additionally, the current flowing into R_{in} in Fig. 4(b) is constant, and it is equal to the current flowing in the feedback, which consists of a memristor and a resistor R_p. This current can be expressed as

$$I_{R_{in}} = \frac{V_{in}}{R_{in}} = I_{R_p}(t) + I_{mem}(t), \qquad (21)$$

where $I_{R_{in}}$ is a constant current flowing into the resistor R_{in}, $I_{R_p}(t)$ and $I_{mem}(t)$ are, respectively, the currents in the resistor R_p and the memristor. Therefore, adaptively amplified voltage pulses can be also achieved by

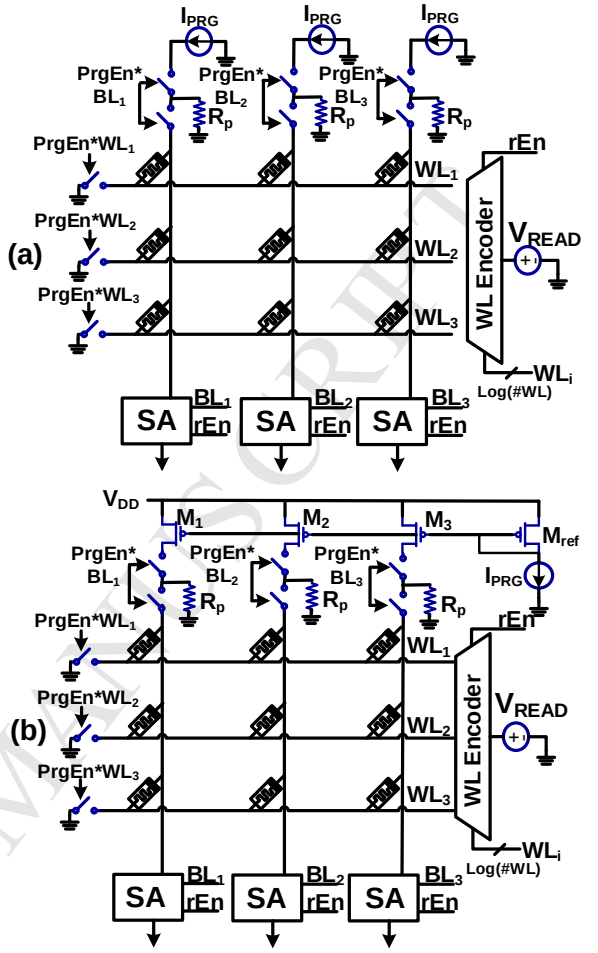

Figure 11: (a) Current programming circuit using multiple ideal current sources and a parallel resistor R_p. (b) Current mirroring-based adaptive programming circuit that uses PMOS transistors [37] and a parallel resistor R_p. Both circuits effectively amplify voltage pulses by maintaining constant current programming with the increasing memristance while programming.

using identical current pulses of V_{in}/R_{in} to program a memristor connected to a resistor R_p in parallel. A possible implementation of current programming is shown in Fig. 11(a), where each opamp is replaced with an ideal current source $I_{PRG} = V_{in}/R_{in}$. The circuit shown in Fig. 11(a) was designed and tested in SPICE simulations, with results similar to those in Fig. 6. Another possible modification is to use simple current mirroring techniques [46], including fewer transistors and less complicated modules for current mirroring to reduce area overhead and complexity of multiple current sources. Current-mirroring-based adaptive programming can be implemented using PMOS transistors as

presented in Fig. 11(b). However, the voltage driving the current mirroring PMOS transistors, V_{DD}, must be sufficiently high to keep the PMOS in the saturation region, *i.e.*,

$$V_{S_iD_i} \geq V_{DD} - V_{G_i} - \left|V_{T_i}\right|, V_{S_iD_i} = V_{DD} - V_{D_i}, \quad (22)$$

where V_{DD} is the voltage driving the current mirroring circuits, V_{S_i}, V_{D_i}, and V_{G_i}, are respectively, the source, drain, and gate voltages of memristors M_i as presented in Fig. 11(b), and V_{T_i} is the threshold voltage of PMOS transistor M_i. Note that all gates of the transistors are connected and the gate voltage of each one equals the source voltage of the reference transistor M_{ref},

$$V_{G_i} = V_{G_{ref}} = V_{S_{ref}} = V_{DD}. \quad (23)$$

Substituting equation (23) into (22), we can formulate the demand on the PMOS transistors to be in saturation mode as

$$V_{D_i} \leq V_{DD} - \left|V_{T_i}\right|. \quad (24)$$

Since V_{D_i} is the voltage dropping on a memory cell (*i.e.*, memristors and a selector in series), and since the programming current is constant, V_{D_i} will increase with the increase of the memristance. Therefore, V_{DD} must be high enough to ensure that V_{D_i} can reach the required voltage amplification (the level necessary to perform adaptive programming), while keeping the PMOS M_i in the saturation mode according to (24), *i.e.*, $V_{DD} = \max(V_{D_i}) + \left|V_{T_i}\right|$.

Different factors can be considered for the design of the peripheral circuits of either of the suggested implementations. The architecture in Fig. 11 eliminates sneak-path currents, and therefore requires no additional components such as transistors (*i.e.*, in 1T1R architecture) to mitigate sneak-paths. Hence, it might be suitable for building fast memory modules, and for embedding in-memory computing, which is discussed in the following section. However, such architectures are limited in capacity due to the increased complexity of routing an increased number of parallel WLs. On the other hand, for large-scale memory modules, the architectures in Fig. 11 might be more appropriate. Moreover, programming structures such as 2TG1R and 4T1R [47], could potentially be applied in MLC ReRAMs for controlling tighter intermediate resistance distributions for high-density modules.

8. AP for In-Memory Computing

Modern applications are limited by the bandwidth between the memory and the processor. To increase performance with this limitation and reduce the need to transfer data between memory and the processor, different architectures have been proposed, including non-von Neumann architectures with in-memory computing [16, 48]. One example is the GP-SIMD architecture, where multi-valued logic operations are performed within MLC ReRAM [16]. Assume an operand A is stored in cell $C_{i,j}$. Then, by applying a number of identical pulses that correspond to the value of operand B on cell $C_{i,j}$ we can store the addition result of the two operands $(A + B)$ in cell $C_{i,j}$, as illustrated in Fig. 12.

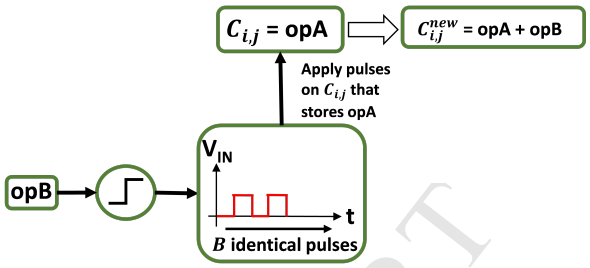

Figure 12: Adding operand B to operand A that is stored in cell $C_{i,j}$.

Since AP allows identical pulse programming in MLC ReRAM, while eliminating the state-pulse dependency and improving FoE, it enables such an architecture. However, logic and architectural challenges should be addressed. These include carry handling, defining a new Instruction Set Architecture (ISA), coherency issues, and other control issues of such memories. In future work, we plan to investigate AP in the context of in-memory computing.

9. Conclusions

Memristors can be programmed to intermediate levels, and not only to HRS and LRS, allowing the design of MLC ReRAM. Due to imperfections in the fabrication process, memory cells deviate from each other and thus limit the number of logical levels stored in each cell. Hence, it is crucial to control the uniformity of level distribution in MLC ReRAM, to increase process variation tolerance and memory capacity.

Adaptive programming (AP) relies on a feedback circuit that amplifies identical voltage pulses according to the current level of the memory cell. AP decreases latency and energy as compared to other programming schemes, with a simpler mechanism. Furthermore, AP reduces the frequency of errors by approximately 50% as compared to other identical pulse programming techniques. Not only does AP improve the memory, but it also has the potential to add computing capabilities

to multi-valued operations, which are useful for in-memory computing.

Acknowledgements

This work was supported by the Viterbi Fellowship at the Technion Computer Engineering Center. The authors would like to thank Chung Wei Hsu and Philip Wong from Stanford University for sharing their experimental results.

Appendix - Selector SPICE Model

```
.SUBCKT SEL_MODEL 1 2
D1 1 2 DF
DZ 3 1 DR
VZ 2 3 2.19
.MODEL DF D ( IS=43.8p RS=35.2 N=1.10
+ CJO=1f VJ=0.750 M=0.330 TT=50.1n )
.MODEL DR D ( IS=8.77f RS=15.1 N=3.00 )
.ENDS
```

References

[1] K.-H. Kim *et al.*, "A Functional Hybrid Memristor Crossbar-Array/CMOS System for Data Storage and Neuromorphic Applications," *Nano Lett.*, vol. 12, no. 1, pp. 389–395, December 2011.

[2] U. Russo, D. Kamalanathan, D. Ielmini, A. Lacaita, and M. Kozicki, "Study of Multilevel Programming in Programmable Metallization Cell (PMC) Memory," *IEEE Transactions on Electron Devices*, vol. 56, no. 5, pp. 1040–1047, May 2009.

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

[4] D. Niu, Q. Zou, C. Xu, and Y. Xie, "Low Power Multi-Level-Cell Resistive Memory Design with Incomplete Data Mapping," *Proceedings of the IEEE 31st International Conference on Computer Design (ICCD)*, pp. 131–137, October 2013.

[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, March 2009.

[6] 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, May 2008.

[7] T. Prodromakis, B. P. Peh, C. Papavassiliou, and C. Toumazou, "A Versatile Memristor Model with Nonlinear Dopant Kinetics," *IEEE Transactions on Electron Devices*, vol. 58, no. 9, pp. 3099–3105, September 2011.

[8] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching Dynamics in Titanium Dioxide Memristive Devices," *Journal of Applied Physics*, vol. 106, no. 7, p. 074508, October 2009.

[9] X. Yang, A. B. K. Chen, B. J. Choi, and I.-W. Chen, "Demonstration and Modeling of Multi-Bit Resistance Random Access Memory," *Applied Physics Letters*, vol. 102, no. 4, p. 043502, January 2013.

[10] 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, February 2012.

[11] C. Xu, D. Niu, N. Muralimanohar, N. Jouppi, and Y. Xie, "Understanding the Trade-Offs in Multi-Level Cell ReRAM Memory Design," *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, May 2013.

[12] F. García-Redondo and M. López-Vallejo, "Self-controlled multilevel writing architecture for fast training in neuromorphic rram applications," *Nanotechnology*, vol. 29, no. 40, p. 405203, July 2018.

[13] A. Bagheri-Soulla and M. Ghaznavi-Ghoushchi, "A high-precision time-domain rram state control approach," *Microelectronics journal*, vol. 74, pp. 94–105, February 2018.

[14] R. Berdan, T. Prodromakis, and C. Toumazou, "High Precision Analogue Memristor State Tuning," *Electronics Letters*, vol. 48, no. 18, pp. 1105–1107, August 2012.

[15] P. Rabbani, R. Dehghani, and N. Shahpari, "A Multilevel Memristor–CMOS Memory Cell as a ReRAM," *Microelectronics Journal*, vol. 46, no. 12, pp. 1283–1290, December 2015.

[16] A. Morad, L. Yavits, S. Kvatinsky, and R. Ginosar, "Resistive GP-SIMD Processing-In-Memory," *ACM Trans. Archit. Code Optim.*, vol. 12, no. 4, pp. 57:1–57:22, January 2016.

[17] H.-S. Wong *et al.*, "Phase Change Memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, December 2010.

[18] D. Apalkov *et al.*, "Spin-Transfer Torque Magnetic Random Access Memory (STT-MRAM)," *Journal on Emerging Technologies in Computer Systems*, vol. 9, no. 2, pp. 13:1–13:35, May 2013.

[19] H.-S. P. Wong *et al.*, "Metal-Oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, June 2012.

[20] S. Kvatinsky, E. Friedman, A. Kolodny, and U. Weiser, "TEAM: ThrEshold Adaptive Memristor Model," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 211–221, January 2013.

[21] H.-Y. Chen *et al.*, "Hfox based vertical resistive random access memory for cost-effective 3d cross-point architecture without cell selector," *IEEE International Electron Devices Meeting*, pp. 20.7.1–20.7.4, December 2012.

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

[23] M.-F. Chang, P.-F. Chiu, and S.-S. Sheu, "Circuit Design Challenges in Embedded Memory and Resistive RAM (RRAM) for Mobile SoC and 3D-IC," *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pp. 197–203, May 2011.

[24] L. Fu *et al.*, "A High Efficiency All-PMOS Charge Pump for 3D NAND Flash Memory," *IEEE International Conference on ASIC (ASICON)*, November 2015.

[25] P. Pavan, R. Bez, P. Olivo, and E. Zanoni, "Flash memory cells-an overview," *Proceedings of the IEEE*, vol. 85, no. 8, pp. 1248–1271, August 1997.

[26] C. Nail *et al.*, "Understanding RRAM Endurance, Retention and Window Margin Trade-off using Experimental Results and Simulations," *IEEE International Electron Devices Meeting (IEDM)*, pp. 4.5.1–4.5.4, December 2016-12.

[27] A. Kalantarian *et al.*, "Controlling Uniformity of RRAM Characteristics Through the Forming Process," *Reliability Physics Symposium (IRPS), 2012 IEEE International*, pp. 6C.4.1–6C.4.5, April 2012.

[28] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.

[29] S. Kvatinsky, M. Ramadan, E. Friedman, and A. Kolodny, "VTEAM: A General Model for Voltage-Controlled Memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, August 2015.

[30] A. M. Abo and P. R. Gray, "A 1.5 V, 10-bit, 14 MS/s CMOS Pipeline Analog-to-Digital Converter," *Symposium on VLSI Circuits*, pp. 166–169, May 1998.

[31] 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*, vol. 44, no. 2, pp. 176–183, February 2013.

[32] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-Path Constraints in Memristor Crossbar Arrays," *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, pp. 156–160, July 2013.

[33] J. J. Huang, Y.-M. Tseng, W.-C. Luo, C.-W. Hsu, and T. H. Hou, "One Selector-One Resistor (1S1R) Crossbar Array for High-Density Flexible Memory Applications," *IEEE International Electron Devices Meeting (IEDM)*, pp. 31.7.1–31.7.4, December 2011.

[34] Z. Jiang *et al.*, "Performance Prediction of Large-Scale 1S1R Resistive Memory Array Using Machine Learning," *IEEE International Memory Workshop (IMW)*, pp. 1–4, May 2015.

[35] 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, May 2010.

[36] 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, May 2012.

[37] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Information-Theoretic Sneak-Path Mitigation in Memristor Crossbar Arrays," *IEEE Transaction on Information Theory*, vol. 62, no. 9, pp. 4801–4813, July 2014.

[38] H. Manem, G. S. Rose, X. He, and W. Wang, "Design Considerations for Variation Tolerant Multilevel CMOS/Nano Memristor Memory," *Proceedings of the 20th Symposium on Great Lakes Symposium on VLSI*, pp. 287–292, June 2010.

[39] S. Lashkare *et al.*, "A Bipolar RRAM Selector with Designable Polarity Dependent ON-Voltage Asymmetry," *IEEE International Memory Workshop*, pp. 178–181, May 2013.

[40] A. Stillmaker, Z. Xiao, and B. Baas, "Toward More Accurate Scaling Estimates of CMOS Circuits from 180 nm to 22 nm," *VLSI Computation Lab, ECE Department, University of California, Davis, Tech. Rep. ECE-VCL-2011-4*, April 2011.

[41] M. Hu, H. Li, and R. E. Pino, "Fast Statistical Model of TiO2 Thin-film Memristor and Design Implication," *Proceedings of the International Conference on Computer-Aided Design*, pp. 345–352, November 2011.

[42] D. Niu, Y. Chen, C. Xu, and Y. Xie, "Impact of Process Variations on Emerging Memristor," *ACM/IEEE Design Automation Conference (DAC)*, pp. 877–882, June 2010.

[43] X. Dong, C. Xu, Y. Xie, and N. Jouppi, " NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, July 2012.

[44] P. R. Gray and R. G. Meyer, " MOS Operational Amplifier Design - A Tutorial Overview," *IEEE Journal of Solid-State Circuits*, vol. 17, no. 6, pp. 969–982, December 1982.

[45] N. Jovanovic, O. Thomas, E. Vianello, B. Nikolić, and L. Naviner, "Design considerations for reliable OxRAM-based non-volatile flip-flops in 28nm FD-SOI technology," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1146–1149, May 2016.

[46] M. Akiya and S. Nakashima, "High-Precision MOS Current Mirror," *IEE Proceedings of Solid-State and Electron Devices*, vol. 131, no. 5, pp. 170–175, October 1984.

[47] X. Tang, G. Kim, P. E. Gaillardon, and G. D. Micheli, "A Study on the Programming Structures for RRAM-Based FPGA Architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 4, pp. 503–516, April 2016.

[48] L. Yavits, S. Kvatinsky, A. Morad, and R. Ginosar, "Resistive Associative Processor," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 148–151, July 2015.