Self-Controlled Writing and Erasing in a Memristor Crossbar Memory

Idongesit E. Ebong, *Member, IEEE*, and Pinaki Mazumder, *Fellow, IEEE*

***Abstract*—The memristor device technology has created waves in the research community and led to the consideration of using the device in multiple avenues. The most likely candidate for early adoption is the nonvolatile memory due to the small cell size (increased scaling potential), increased density as compared to flash, and ability to stack these devices in a crossbar structure. This paper analyzes the feasibility of a memristor memory and introduces an adaptive read, write, and erase method that may be used to realize a more resilient memory system in the face of low yield in the nanotechnology regime. The proposed method is evaluated in simulation program with integrated circuit emphasis (SPICE) and a hand analysis model is extracted to help explain the sources of power and energy consumption. Finally, the power metrics are compared to flash memory technology, and the memristor memory is shown to have an energy per bit consumption about one-tenth that of flash when programming, comparable to flash when erasing, and about one-fourth of flash when reading.**

***Index Terms*—Memristor, resistive random access memory (RAM).**

I. Introduction

MEMORY is an indispensable part of electronic devices today. Multiple variants of memory exist and have led to a myriad of niches for multiple memory concoctions. These memory concoctions reside in an ever-changing technological domain allowing for the categorizing of different memory types: volatile versus nonvolatile; fast versus slow; low capacity versus high capacity; and cheap versus expensive. Memory examples include random access memory (RAM), flash, hard drives, and optical disks. Flash memory dominates the nonvolatile memory market today for handheld and battery-operated devices.

Since HP's identification [1] of Chua's memristor [2], multiple applications for the device have been proposed ranging from memory [3] and reconfigurable logic [4] to neuromorphic learning [5] and secure communication [6]. From all the applications, the most promising with respect to product development is the digital memory utilizing memristors as storage elements. A new paradigm with respect to memory is necessary for the continued growth in density of nonvolatile memory for anticipated growth in petascale and exascale computing. The memristor's simple structure, small size compared to transistors, and nonvolatility make it a viable candidate for next-generation memory technology. Memristor memory is a subset of resistive memory since logic states are encoded in the memristor's resistance. Even though resistive memory is a more general term, some problems associated with resistive memory in a crossbar array are also characteristic to the memristor memory. The difference between resistive memory and memristor memory lies in the fact that memristors have a pinched hysteresis loop at the origin, while the more general term, resistive, includes devices such as the one in [7], which do not possess this trait. Resistive memory in essence comprises a lump of devices with differing resistance-change mechanisms. The method introduced in this paper, hence, may not be applicable to all resistive memory devices, but it is definitely advantageous to memristor memory systems.

The memristor memory presents a solution to difficulties encountered beyond CMOS scaling, but it also introduces various complications to realizing this memory system. The patent database provides a myriad of methods to deal with difficulties (resistance drift, nonuniform resistance profile across the crossbar array, leaky crossbar devices, etc.) that arise from working with these resistive memory elements. These difficulties (problems) are addressed within the database by using correcting pulses to mitigate effect of resistance drift due to normal usage [8]; using a temperature-compensating circuit to counter resistance drift due to temperature variation [9]; using an adaptive method to read and write to an array with nonuniform resistance profile [10]; and introducing diodes [11] or metal–insulator–metal (MIM) diodes to reduce leaky paths within the crossbar memory array [12].

With every proposed solution to counter a problem, there are drawbacks that need to be considered. This work exposes a view that will lead to the realization of memristor-based memory in the face of low device yield and the aforementioned problems that plague memristor memory. Section II briefly introduces the memristor, isolating diode, and crossbar modeling employed. Section III describes the reading, writing, and erasing methodology. Section IV shows the simulation results. Section V explains the results. Section VI provides concluding remarks.

II. Memristor and Crossbar Modeling

The memristor model used for simulation is based on the nonlinear drift model with the window function $F_p(2)$ as defined in [13]. The doped-region width w is modulated according to (1) with the window-function definition expressed in (2). For the SPICE simulation, the memristor model was implemented as a functional block in Verilog-A with parameter $p = 4$, memristor

Manuscript received March 21, 2011; accepted August 18, 2011. Date of current version November 9, 2011. The review of this paper was arranged by Associate Editor J.-P. Leburton.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: idong@umich.edu; mazum@eecs.umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNANO.2011.2166805

width $D = 10$ nm, and dopant mobility $\mu_D = 10^{-9}$ cm^2/V·s:

$$\frac{dw}{dt} = \frac{\mu_D R_{\rm ON}}{D} i(t) F\left(\frac{w}{D}\right) \quad (1)$$

$$F_p(x) = 1 - (2x-1)^{2p}. \quad (2)$$

Diodes are used for crossbar isolation of individual devices in accordance with [12]. For simulation, the memristor is in series with a bidirectional diode model, representative of the MIM diode, presented as

$$I_{\rm Diode} = I_0(e^{qV_D/nkT} - 1). \quad (3)$$

Overall, the simulation parameters for the diodes were: $I_0 = 2.2$ fA, $kT/q = 25.85$ mV, V_D is dependent on applied bias, and $n = 1.08$. A P-N diode model is used because it provides a weaker isolation than actual MIM diodes. Hence, if the proposed adaptive method works with the P-N diode configuration, then it will work better with the actual MIM configuration that depends on tunneling currents and provides better isolation than P-N diodes.

Nanowire modeling for simulation is a distributed pi-model, but for hand calculations, a lumped model is used. From [14], the nanowire resistivity follows:

$$\frac{\rho}{\rho_0} = 1 + 0.75 \times (1-p)\left(\frac{\lambda}{d}\right) \quad (4)$$

where ρ_0 is the bulk resistivity, d is the nanowire width, and λ is the mean free path. The nanowire-recorded value used for the simulation was: 24 $\mu\Omega$·cm for 4.5-nm thick Cu. Following a conservative estimate, the nanowire resistance was chosen to be 24 kΩ in total. Using a nanowire capacitance of 2.0 pF·cm^{-1}, the nanowire modeling was made transient complete.

III. Approach

A. Top-Level Memory

In Section II, we had set up and described the contents of the memory crossbar array (MCA) depicted in Fig. 1. This section will help expand upon the memory architecture envisioned and provide more details with respect to the other components, i.e., the muxes/demuxes, read circuitry (RC), and the data divisions. In this section, we will also introduce the reading, writing, and erasing schemes utilized in the envisioned memory system.

Fig. 1 shows the top-level block diagram and the connections between the already-explained crossbar array and the periphery circuitry. The row- and column-address signals allow a selected row or column to be transparent to either the RC or the data sections.

The nature of the muxes may prove to make design more difficult due to the stringent requirements of their functionality. These requirements do not affect the muxes controlled by the reverse polarity (RP) signal; these muxes are simpler since they are essentially transmission gate muxes that switch between two paths. For the row- and column-address muxes, the mux requirements extend beyond switching paths for unselected and

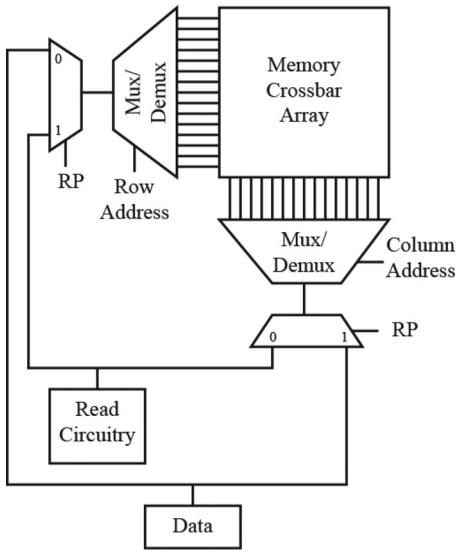

Fig. 1. Top-level block diagram.

selected lines. Our preliminary simulations agree with the results from [15] proposing active bias for unselected lines (columns and/or rows). When a line is unselected, a reference bias must be set on all the unselected lines thereby limiting the leakage paths that may affect read and write integrity. For more details on this problem, refer to [16], where the authors discussed in detail the effect on noise margin of floating the unselected lines in a resistive memory.

In this implementation, the selected and unselected lines have two different references corresponding to when the memory is in use and when the memory is not in use. When in use, the unselected lines are held at $V_{\rm REF}$ voltage, while when not in use, the lines are grounded. The selected lines pulsate between $V_{\rm REF}$ and $V_{\rm DD}$ when memory in use but is held to ground when memory is not in use. The signal flow is unidirectional from data, through an RP mux, through a mux/demux, through the MCA, through another mux/demux, then another RP mux, and finally to the RC. The signal-flow direction is controlled by which RP mux is connected to the RC and by which RP mux is connected to data.

Data is a small driver that asserts $V_{\rm DD}$. The length of time $V_{\rm DD}$ is asserted is controlled by timing circuits that determine when to open the signal path from data to RC. The RC block is essentially a generic block that implements the flow diagram represented in Fig. 2, specifically, the "Calculate δ" and the steps that lead to determining the logic state of the selected memristor device. This signal flow is used to avoid negative-pulse-generation signals as seen in [17] and [18].

B. Read, Write, and Erase Operations

This section delves into the operations of the RC division with respect to the flow diagram in Fig. 2. Fig. 2(a) shows the decision process for a read, while Fig. 2(b) shows the decision process for a write or erase operation. The write and erase operations are the extensions of a single-cycle read operation. The double-cycle read is given in the flow diagram, and this is dubbed double

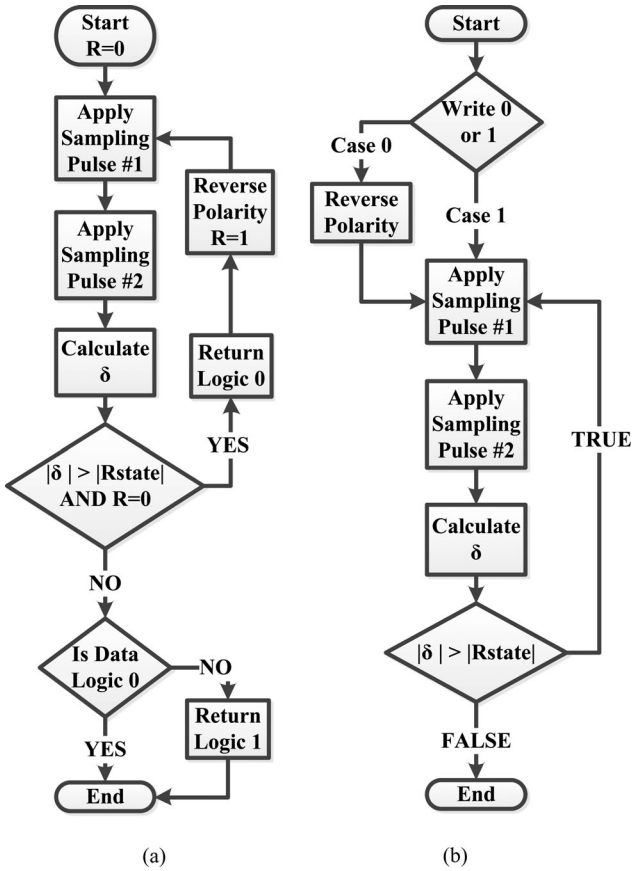

Fig. 2. (a) Read flow diagram. (b)Write/erase flow diagram.

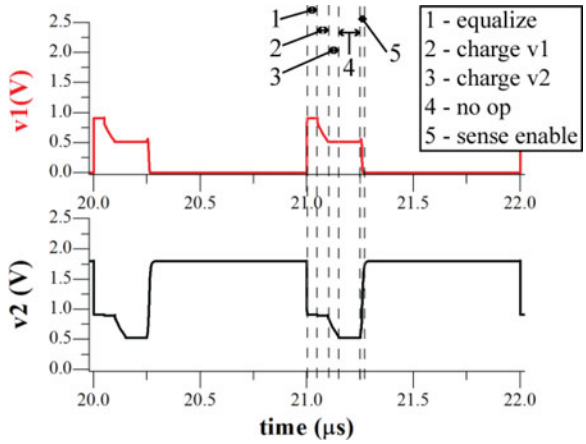

Fig. 3. Memory cell read operation showing the different phases of read: equalize, charge v1, charge v2, no op, and sense enable.

cycle because the memristor reads in one direction and, then, reads in the other direction to restore state if necessary. The read process is designed this way in order to prevent or limit read disturbance. Since each memory device in the crossbar array is different, the chosen pulses utilized for the read may cause destructive reads thereby requiring a data refresh after read. The refresh process is essentially built into the read just in case it is necessary.

Referring back to Fig. 2(a), we apply bias to the memristor to sample its current value (apply sampling pulse 1), and then, we apply another bias to the memristor to sample its value again (apply sampling pulse 2), and finally, calculate δ. δ signifies the amount of change that has occurred within the memristor between the two sampled pulses. The pulses are chosen in a manner that will change the conductance of the memristor. Depending on the magnitude of δ, the read circuitry will return either a "Logic 0" or a "Logic 1." The definitions of both the "Logic 0" and "Logic 1" states depend on the designer. In one state, the sampling pulses push against an upper (lower) limit, while in the other state, the sampling pulses move the memristor in a direction opposite to its current state. In the latter case, a correction is necessary if considering that each memristor is different within the crossbar array. The pulses used will disturb memory state based on both the location of memristor within the crossbar array and the low/high resistance boundaries of specific memristors. The unknown memristor resistance response to the applied pulses puts a requirement of a loop back requiring a polarity reversal.

The read process described in Fig. 2(a) is extended to create Fig. 2(b). The goal of the latter figure is to reuse circuitry for the erase and write operations. The erase operation is defined as taking the memristor from a "Logic 0" to a "Logic 1," while the write operation changes the memristor from a "Logic 1" to a "Logic 0." These states can be interchanged depending on definition, as long as the definition is consistent across the read, write, and erase operations.

The adaptive write process is similar to that in [19]. While the presented process is a discrete process that requires multiple steps, the process in [19] continuously changes the memristor until a latch stops the write process. This method is appropriate for single devices, but using a control to stop an applied bias may be tricky to implement in a crossbar because signal delays will come into play. The delays may cause an overprogramming or overerasing of a device, or even overdisturbing unselected devices.

The advantages of reading, writing, and erasing using the proposed scheme in Fig. 2 includes: tolerance to crossbar-variation resistance; adaptive method to write and erase a crossbar memory; and circuitry reuse for read, write, and erase. The evaluation of the method in the following sections will strive to provide evidence of these claims. Before diving into the evidence, a circuit model of the previously shown flow diagrams must be presented.

Fig. 3 shows the different tasks (equalize, charge v1, charge v2, no op, and sense enable) that compose a read. The circuit that produces these signals is shown in Fig. 4 to make sense of the different tasks. Two sampling signals, i.e., φ_1 and φ_2, control the conversion of current-to-voltage samples on capacitors C_1 and C_2. But before any sampling, an equalize operation is performed to balance the charges on both capacitors by asserting EQ signal high. Once the signals are sampled, then the sense-enable operation is performed by first asserting NS high and, then, later PS high. The sense amplifier in Fig. 4(b) is modified from the sense amplifiers found in the literature. The amplifier

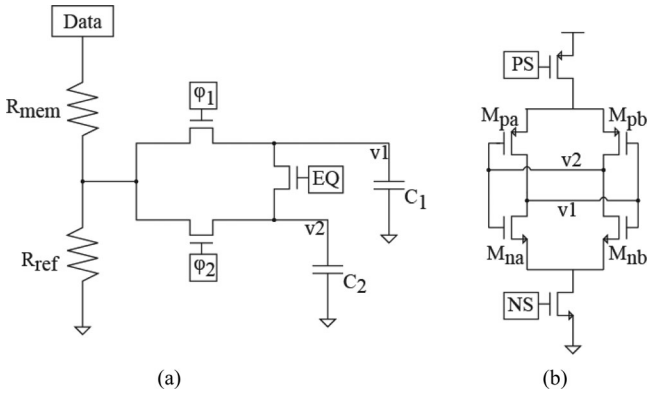

Fig. 4. Read sense circuitry. (a) Sampling circuit that converts current through R_{mem} to voltage. (b) Sense amplifier that determines high- or low-resistive state.

is purposefully made unbalanced to produce a default output of low resistance.

The unbalanced attribute of the sense amplifier can be achieved in multiple ways, but the chosen method in this implementation is to make the W/L ratio of both M_{pa} and M_{pb} 320 nm/180 nm; the W/L ratio of M_{na} 1 μm/500 nm; and W/L ratio of M_{nb} 1.2 μm/500 nm. The NMOS devices are unbalanced, while the PMOS devices are balanced. The transistor controlled by NS has a ratio of 280 nm/180 nm, while the one controlled by PS has 400 nm/180 nm. R_{ref} is an 80-kΩ resistor, while R_{mem}'s default value is expected to vary from 20 kΩ to 20 MΩ.

IV. Simulation Results

The simulation approach consists of considering different memory conditions on a 16×16 array. The device of interest is situated in the center of the array, but all verifications were done with a worst-case device at the corner with minor changes in the results. The crossbar array unless specified otherwise contains all memristors with the ability to change states.

A. High-State Simulation

In high-state simulation (HSS), the memristor crossbar array has all devices initialized to a high conductive state (the worst-case scenario). The device of interest to be written to has a resistive range between 20 kΩ and 20 MΩ, and its initial resistance is ∼18 MΩ. The device accessed for the write operation is located at the center of the array (eighth row, eighth column). Fig. 5 provides a sample number for read cycles necessary to perform the write operation.

Fig. 5(a) shows the number of cycles required for a write, while Fig. 5(b) shows the change in memristance of the accessed device in each read cycle. Each read operation provides the device-state feedback, and the device only changes from high resistance to low resistance when the device is written to its lowest resistance level, i.e., 20 kΩ. The number of read cycles necessary to write in this case is ∼21. The signals v1 and v2 presented in Figs. 3 and 4 are appropriately renamed to help facilitate the understanding of the simulation results.

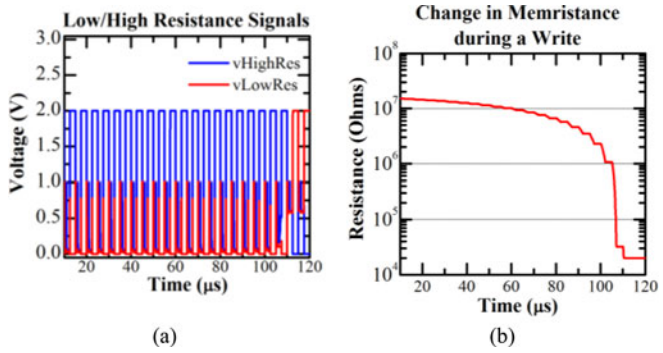

Fig. 5. Simulation results writing to an RRAM cell. (a) Low/high-resistance signals. (b) Memristance high-resistance to low-resistance switch.

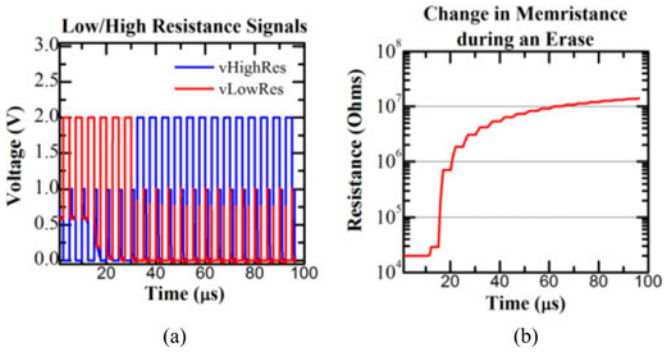

Fig. 6. Simulation results erasing an RRAM cell. (a) Low/high-resistance signals. (b) Memristance low-resistance to high-resistance switch.

"vHighRes" and "vLowRes" are the logically renamed signals to denote when the device of interest is in a high-resistance state and a low-resistance state. When the signal vHighRes is high, the memristor is in a high-resistance state, but when vLowRes is high, the memristor is in a low-resistance state. Both vHighRes and vLowRes are always opposites of each other in the sense-enable phase.

Fig. 6(a) shows the number of cycles required for an erase, while Fig. 6(b) shows the change in memristance of the accessed device. Just like the write cycle, the erase cycle is performed through read operations. The erase cycle takes six read cycles to go from a low-resistive state to a high-resistive state. The sense amplifier recognizes the switch to a high-resistive state when the resistance is about 4.21 MΩ. This implies that during memory operation, the number of read operations necessary for a write after an erase may be different. And this adaptive method will prevent any overerasing or overwriting (overprogramming).

B. Background-Resistance Sweep

In the background-resistance-sweep (BRS)-simulated state, the background resistance for all devices is swept from 20 kΩ to 20 MΩ. The device of interest is kept the same as in the HSS case: its resistance range is from 20 kΩ to 20 MΩ. The goal of the simulation is to show the effect of current memory state on reading, erasing, and writing to a selected memristor. Figs. 7 and 8 show the simulation result for a broad spectrum (20 kΩ, 200 kΩ, 2 MΩ, and 20 MΩ), from top to bottom. Since tuning

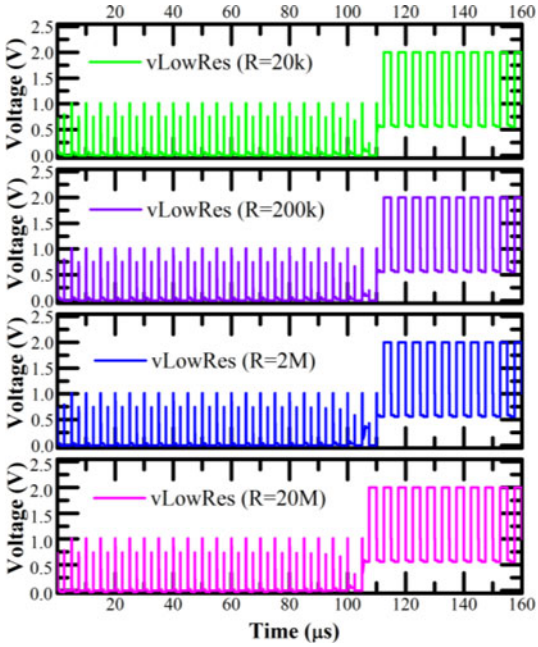

Fig. 7. Writing in the BRS case showing that resistance background has the minimal effect on the number of read cycles required for a write.

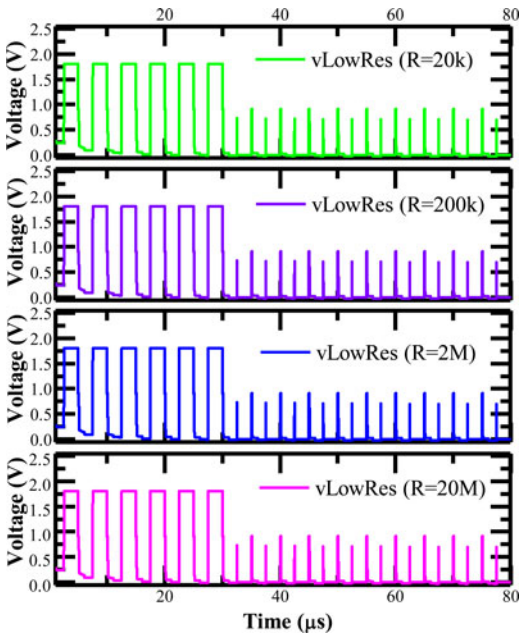

Fig. 8. Erasing in the BRS case showing that resistance background has the minimal effect on the number of read cycles required for an erase.

memristors to specific resistances is a time-consuming process, the background resistance for all devices is achieved with static resistors. Fig. 7 shows the simulation results for the write case, while Fig. 8 shows the simulation results for the erase case.

From Fig. 7, the starting resistance is about 16 MΩ, and ∼21 read operations are necessary for a write. In the 20-MΩ case, one less read is required. The simulation results show only vLowRes signal for clarity (vHighRes is its opposite as shown earlier in Figs. 5 and 6).

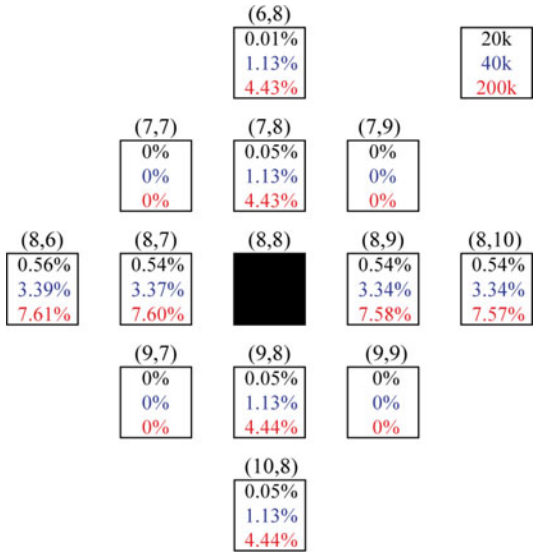

Fig. 9. Percent change in unselected devices during an erase for different minimum resistances.

The BRS experiment is performed for the erase case to show that using the memristor, with proper diode isolation, a similar result is obtained. The same number of read cycles is necessary to erase the memristor in all four background-resistance sweeps.

Another concern besides the background resistance is the effect of reading, writing, and erasing on unselected devices. A BRS experiment was performed but instead of using static devices around a memristor, the memory array was composed of all memristors with background resistances around 20, 40, and 200 kΩ. The maximum resistance for all devices still remained at 20 MΩ. Fig. 9 provides the results for the change in unselected devices during an erase operation.

In Fig. 9, the larger the minimum resistance, the larger the percent change undergone by the unselected memristors. This simulation hints that the larger the spread between the minimum and the maximum resistance, the less likely unselected memristors will change. Another factor that may contribute to the results of Fig. 9 is that the lower the minimum resistance is compared to the resistance of an OFF diode, the less likely the memristor will change. This is because of the voltage divider set up by the memristors in series with the diode whereby most of the voltage drop is on the diode thereby causing very little voltage drop on the unselected memristor.

C. Minimum-Resistance Sweep

For the minimum-resistance-sweep (MRS) case, the resistance range for the memristor of interest is modified. Since the BRS case has shown that the background resistance is really no factor with proper diode isolation, the HSS simulation conditions are used whereby unselected devices are initialized to low resistance and may change during the writing operation. Fig. 10 shows a coarse spread of low resistances and the number of read cycles necessary to complete a write. This result suggests that with the set pulse duration for sampling, there exists a continuum on the number of read cycles necessary before a write

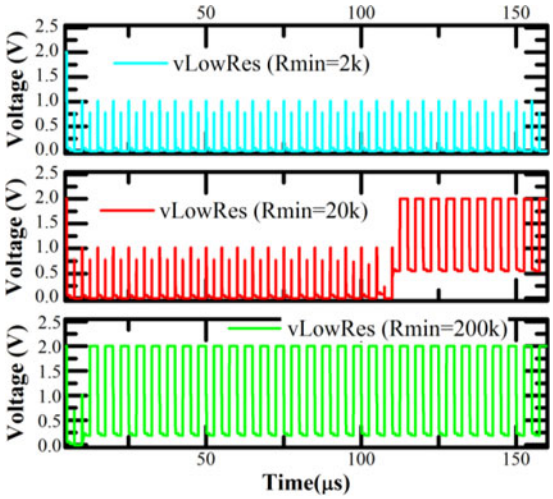

Fig. 10. Writing to memristor devices with the same high-resistive state but varying low-resistive states (coarse spread). The minimum resistance affects the number of read cycles necessary before a write occurs.

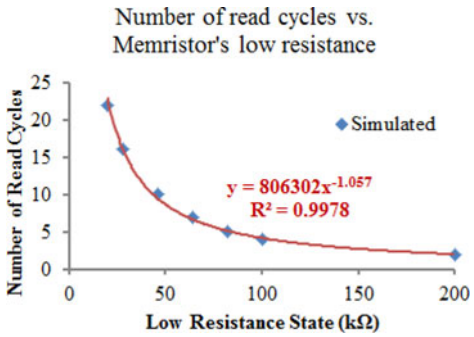

Fig. 11. Writing to memristor devices with the same high-resistive state but varying low-resistive states. The larger the minimum resistance, the lower number of read cycles necessary to reach the low-resistive state.

occurs. The farther the lowest resistance is from 20 MΩ, the more the number of read cycles necessary for a write to occur. In the 2-kΩ case, the switch to a low-resistive state does not occur. In the 20-kΩ case, the switch to a low-resistive state occurs after ∼21 read cycles, and in the 200-kΩ case, the switch to a low-resistive state occurs after one read cycle. This trend implies that the current parameters chosen for sensing may be limited to the range currently provided. For the cases, where the low-resistive state is greater than 200 kΩ, the sensing circuit might only give vLowRes as high. The sensing resolution takes a hit here but this can be adjusted by using a shorter pulse width.

The implication of an upper end only means that for devices with low-resistance states closer to their high-resistance states, shorter sampling pulses will need to be used in order to detect the memory state. Shorter pulses will provide the resolution necessary to avoid overwriting. Fig. 10 might show a coarse sweep, but Fig. 11 shows a finer sweep of the minimum resistance. The trend mentioned earlier holds true when the low-resistance state is varied from 28 to 100 kΩ. As the low-resistance-state value increases, the number of pulses required to reach this value decreases.

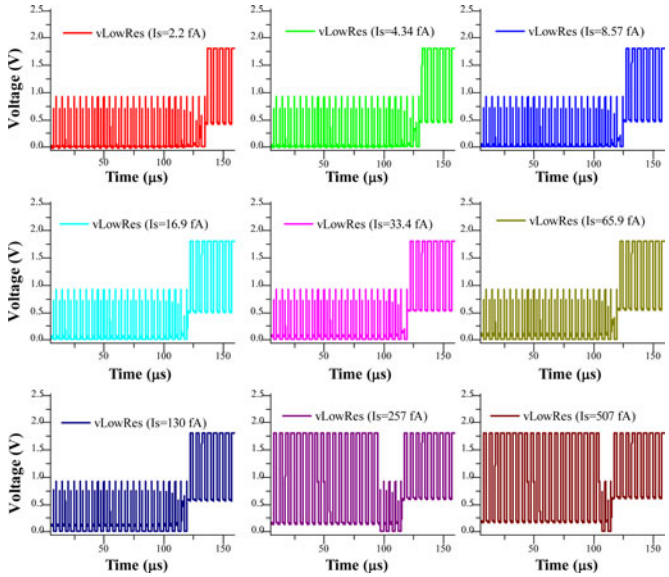

Fig. 12. Writing under different diode-leakage conditions (from left to right: 2.2, 4.34, 8.57, 16.9, 33.4, 65.9, 130, 257, and 507 fA) to show that under heavy leakage, the read/write circuitry fails to correctly determine logic state of memristor.

D. Diode Leakage Current Simulation

The goal of this simulation is to determine how much diode leakage the 16 × 16 network's sensing scheme can handle. The graphs shown in Fig. 12 depict multiple read cycles under different diode saturation currents I_S. The saturation currents going from left to right are: 2.2, 4.34, 8.57, 16.9, 33.4, 65.9, 130, 257, and 507 fA. For the first seven I_S values, the sensing scheme works as expected. For the lowest saturation current, i.e., 2.2 fA, it takes about three more read cycles for a write to occur as opposed to the highest saturation current, i.e., 130 fA. The sensing scheme fails for the 257-fA case and 507-fA case.

In Fig. 12, the higher leakage cases actually switch the memristor device state more quickly than the lower leakage case. The failed cases (257 and 507 fA) do not signify a change in memristor characteristic behavior, but they signify a drawback in the sensing mechanism. This view is supported in the simulation results of Fig. 13. The memristor responses to the pulses provide the same general shape; therefore, the sensing method should be able to determine the resistive state. The high-leakage cases take the memristor to a low-resistive state quicker than the low-leakage cases and this is verified also in the memristance profiles.

A redesign of the sensing circuit can overcome this drawback and only suggests that the circuit only responds to certain limits. By resizing the sense amplifiers, a better leakage range can be accommodated at the cost of lower precision.

E. Power Calculations

For hand analysis, a lumped wire model is used for the nanowire as shown in Fig. 14, but for simulation, a distributed pi-model is used. The capacitance C_N is in the femtofarad range, while C_{M1} is in the attofarad range. The capacitors of

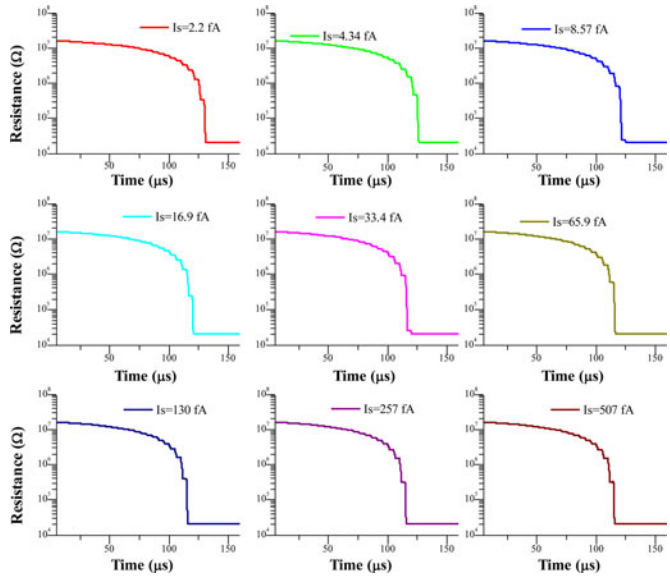

Fig. 13. Memristor changes under the different leakage conditions showing that the read/write failure in Fig. 12 is not because of the characteristic deviation but because of sensing methodology drawback.

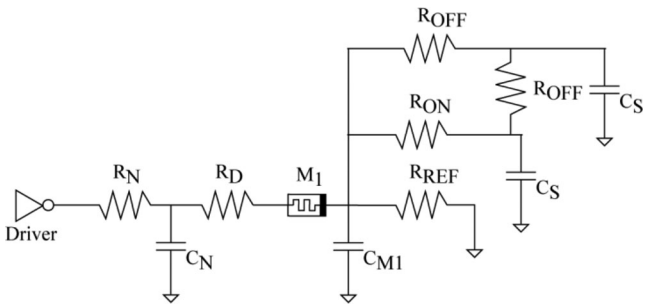

Fig. 14. Equivalent circuit schematic showing the components considered in power analysis (note that series diode $R_D \ll M_1$).

interest that contribute most to the transient behavior of the chosen method are the C_S capacitors that have capacitance in the hundreds of femtofarad range.

Using a Delta–Wye conversion and ignoring some capacitors, the time constants related to the OFF and ON resistance paths are derived. The small capacitors, i.e., C_N and $C_{M\,1}$, are ignored in this analysis for sake of simplicity since they are much smaller than C_S. The ON and OFF paths relate to the switches in Fig. 4 that are controlled by the sampling signals, i.e., φ_1 and φ_2, and the EQ signal.

There are four noticeable sources of power consumption. The first comes in the form of power dissipated by the resistive nature of the nanowires, transistors, and memristors. The second comes in the form of dynamic power needed each cycle due to capacitances that charge and discharge. The third source comes from nonideal isolations and leakage, i.e., diode leakage in the nanocrossbar array or OFF transistor leakage. The last source of power comes from the static and dynamic nature of the driving circuitry used to drive the crossbar array. The third and fourth sources of power severely depend on implementation and will not be considered in the following analysis; note though that with CMOS scaling, these may dominate future power consumption.

The power analysis is done for one complete read cycle, and depending on the amount of read cycles necessary for a write or an erase, the equations can be iterated through N cycles to estimate the power for the necessary number of cycles.

1) The worst-case C_S charging and discharging energy: $C_S(V_{\mathrm{REF}})^2$.
2) The worst-case energy dissipated in the resistor reference: $(I_N(M_1))^2 \cdot (R_N + M_1) \cdot t_s + ((V_{\mathrm{REF}})^2/R_{\mathrm{REF}}) \cdot t_s$, where t_s is the average time for which the resistor combination is under bias.

a) Programming and Erasing Sequence: During programming and erasing, the value of M_1 changes with the applied bias. For hand analysis and verification of the programming and erasing sequence, a model is necessary that will account for memristance change from high to low and from low to high depending on the sample voltage pulses. The change in memristance is discretized in (5) through N read cycles necessary for programming or erasing:

$$M_T = R_0\sqrt{1 - \frac{2 \cdot \eta \cdot \Delta R \cdot \phi(t)}{Q_0 R_0^2}} \cong R_0\sqrt{1 - \frac{2 \cdot \eta \cdot \Delta R \cdot \sum_{n=1}^{N} v_n \cdot t_s}{Q_0 R_0^2}}. \quad (5)$$

The memristance values over time follow the definition of M_T in (5). Here, M_T is the total memristance; R_0 is the initial resistance of the memristor; η is related to applied bias ($+1$ for positive and -1 for negative); ΔR is the memristor's resistive range (difference between the maximum and the minimum resistance); $\phi(t)$ is the total flux through the device; Q_0 is the charge required to pass through the memristor for the dopant boundary to move a distance comparable to the device width; and v_n is the voltage across the memristor.

For programming, the adaptive method registers a change from high resistance to low resistance when the memristor hits 20 kΩ. For erasing, the change from low resistance to high resistance occurs around 4.21 MΩ. Iteratively, the power and energy is determined using constant time steps of t_s.

For the simulation/hand analysis, the values used are: $R_{\mathrm{REF}} =$ 80 kΩ, $R_N = 26$ kΩ, $C_S = 320$ fF, $t_s = 2$ μs, and $M_1 = 18$ MΩ for a high-resistive state and 20 kΩ for a low-resistive state. The V_{DD} value for this simulation was chosen as 1.8 V and adjusted down to 1.1 V to account for drops on the MIM diode. With these parameters, the power consumed for each read cycle in the low-resistive state is 9.68 μW, while the power consumed in the high-resistive state is 0.07 μW.

For the SPICE simulated case, the power consumed for each read cycle in the low-resistive state is 10.5 μW, while the power consumed in the low-resistive state was 0.67 μW. The values for the low-resistive state are similar to the calculated but the value for the high-resistive state is a great underestimation (89.6% error)!

TABLE I
POWER AND ENERGY SUMMARY

Power (μW)

	Calculated	Simulated	% Error
Read high resistance	0.66 μW	0.67 μW	-1.49
Read low resistance	10.27 μW	10.5 μW	-2.19
Program*	23.83 μW	35.9 μW	-33.62
Erase**	13.21 μW	15.3 μW	-13.7

Energy per bit (pJ/bit)***

	Calculated	Simulated	% Error
Read high resistance	1.32	1.34	-1.49
Read low resistance	20.55	21	-2.14
Program*	47.67	71.8	-33.62
Erase**	26.41	30.6	-13.7

*Twenty six read cycles necessary for a write in the simulation, while this number is less in hand calculation.
**Calculated changed to match number of cycles necessary to exceed 4.21 MΩ and not the number of cycles necessary to erase device to ~20 MΩ.
***2 μs total pulse width used for each read cycle.

b) Inclusion of the Leakage Estimation: The high-resistive state is definitely a victim to the leakage power. The simulation in this study is done in a low-resistive memory state to account for the worst-case condition. In this memory state, the measured leakage value for devices in the selected rows and selected columns is around 20 nA each. In our 16×16 array, this accounts for 30 devices biased to around 0.9 V (lower than the MIM diode threshold); therefore, the leakage increases due to the applied bias. The diodes are modeled with two P-N diodes in series for the worst-case performance, while the actual MIM characteristics will be better.

In order to estimate the energy more efficiently, this leakage power must be accounted for. This was done by using the diode equation in (3), with $I_0 = 2.2$ fA, $kT/q = 25.85$ mV, $V_D = 0.45$ V (0.9 V divided equally by two identical P-N diodes) and $n = 1.08$, $I_{\mathrm{Diode}} = 22$ nA. Assuming each path on the selected rows and columns takes a diode current of this magnitude; then, the total power consumed by leakage in the 16 × 16 array is 30×22 nA $\times$ 0.9 V $= 0.59$ μW. Adding this value to the hand-calculated values in the previous section gives better agreement with the simulation in both resistive states: 10.27 and 0.66 μW.

To summarize, the energy per bit for the memristor memory compared to flash looks very promising. The numbers from flash include the periphery circuitry and driving circuitry. Most energy consumption in flash is usually attributed to the charge pumps, which are unnecessary in the resistive memory case. In flash memory product comparison, the lowest read energy for single-level cells is 5.6 pJ/bit, program energy 410 pJ/bit, and erase energy 25 pJ/bit [20]. These values are from different single-level cells (one product could not boast to be the lowest in all categories). The read and erase energy per bit for the resistive memory is given in Table I. There is potential of reducing the program energy significantly by shifting to resistive memory technology. The erase energy between this technology and flash are similar, and the read energy depends on the state of the memristor being read.

V. DISCUSSION

The resistive RAM (RRAM) is a structure that strives on the isolation provided from one cell to the next cell. The ability to selectively access one device without disturbing the other is the most vital trait of the technology. The results from the diode leakage current (DLC) simulation show the vulnerability of sensing in the resistive memory when the leakage current is too high. One way to combat this effect is to allow for an adjustable reference resistor and design for specific leakage tolerance. The BRS results showed that as long as the diode isolation was intact, the memory state does not dominate device-state sensing. In essence, the proposition of more tolerable sensing methods does not eliminate the need for tighter device processes with respect to isolation.

The method proposed provides a sensible way to deal with errors (defects) in the crossbar structure. Errors can be classified in three ways: 1) the memristor is in a stuck-open state; 2) the memristor is in a stuck-closed state; and 3) the lower bound or upper bound resistance targets are not met. In the first two errors (stuck open or stuck close), an attempt to write the opposite data to the memristor will fail. In either case, as long as the memristor is static, the write methodology will only attempt the write process once. The read process will always produce a Logic 1 as defined in the flow diagram in Fig. 2(b). The stuck-open or stuck-close case will not take multiple write cycles in order to determine if the memristor is functional. To determine if the device works or not, a read in one direction is performed, an opposite data write is tried (again lasting only one read cycle due to the static nature of the failed device), and a read verify is performed. If both reads yield the same result, then the device is nonoperational. This method removes the guesswork from setting hard thresholds and setting the maximum write tries before a memory-storage cell is deemed defective.

The defective nature of a stuck-open or stuck-closed cell is different from a device that misses the target high and low resistances for memristor devices. These devices behave in a way that exhibit hysteresis but may have larger or smaller ratios of the resistance in the high state to the resistance in the low state compared to the design target. Since the proposed method does not deal directly with the absolute resistance values, the exact extremes of the resistance of a certain device is not of interest. Resistance extremes are dealt with in ratio (see Fig. 10). The larger the range between the high- and low-resistive states, the more the number of read cycles necessary to perform a write or erase operation. Also, depending on resistance range, the pulse widths used for the design may not be enough to distinguish high and low states. For example, in Fig. 10, any low level greater than 200 kΩ does not provide enough separation between the high- and low-resistive states. The chosen 1-μs pulse widths would already change the device state from one extreme to another during a read operation. The analyses done in this study examines the memory limitations for a chosen pulse width, but the values presented can be improved upon with shorter pulses (<1 μs) based on the improved memristor switching performance.

The advantage of using this method for read/write is to combat the effects of process variation within the crossbar structure. The exact low level does not matter except that the level is within operational limits imposed by the 1-μs pulse. The nature of the low level and high levels of memristive devices to change during operation requires that the sensing method take this into account. During operation, as long as the pulses do not change

the memristor device to an extreme, then a device that may have been deemed a failed device under another sensing scheme is salvaged for further use. This method provides an insightful scheme to combat the effects of resistance drift as memristors' absolute extreme resistances change over their lifetimes.

The power and energy numbers in Table I show some disagreement between calculated and simulated values. Eliminating the assumptions made due to the low time constant values for the different capacitor paths may help agreement. Essentially, the storage capacitors, although their access transistors are in the OFF state, are leaking and charging depending on the cycle presented by data. Also, the peripheral circuitry consumes power not included in the calculated values. Considering that the same driving circuitry is used to drive the memristor in both its high- and low-resistive states, the low current achieved in the high-resistive state suggests that the time constants of the OFF and the ON paths have similar power characteristics, which accounts for 0.01 μW. However, in the low-resistance state, the OFF and ON paths have differing power profiles leading to 0.23 μW disagreement between simulated and calculated.

The program and erase numbers have a larger error differential because two different models are used to determine the weight change in the memristor. In the calculated case, the weight change is determined through an approximated linear diffusion model whereby boundary effects are not taken into consideration. In the simulated model, boundary effects are modeled with a window function, which is why when the device is in a low-resistive state at a boundary albeit high current, the memristance does not change as drastically as predicted by the linear model.

The current method proposed takes into account problems that may be more pronounced in the higher dimension grid, i.e., 4-KB block size as used in many commercial flash devices. The resistive nature of the nanowire will be more pronounced for devices not very close to the driver. This method of determining memory state adjusts to the resistive drops that may be made when the nanowires are more resistive than expected. The problem that may affect a larger memory size is excessive voltage drops, which would require tuning the voltage level to accommodate all devices in the crossbar array. Devices far from the drivers will essentially take longer to write or erase compared to devices closer to the driver. Essentially, an adaptive read, write, and erase method allows for a more flexible process technology and will enable the adoption of the memristor memory sooner since devices that do not meet high- and low-resistance criteria may still be used with confidence.

VI. Conclusion

The showcased memristor memory extols the advantages of using the new technology in memory applications. The method of achieving the read, write, and erase relate adaptively to each memristor device thereby allowing for the increased yield when it comes to using devices that have differing high to low resistance range. The memristor memory also exhibits lower power and energy consumption when compared to flash memory. Unfortunately the proposed method cannot be directly applied to the multibit memory since this method depends on writing the memristor to an extreme. New methods will need to be devised that will allow for reliably writing to the device in the multibit case, as well as perform flash like operations, such as block erasures. The latter is not necessary, but it would improve the operation per bit statistics when it comes to the power and energy consumption.

References

[1] 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.

[2] L. O. Chua, "Memristor—The missing circuit element," *IEEE Trans. Circuit Theory*, vol. CT18, no. 5, pp. 507–519, Sep. 1971.

[3] J. J. Yang, M. D. Pickett, X. M. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, vol. 3, no. 7, pp. 429–433, 2008.

[4] Q. Xia, W. Robinett, M. W. Cumbie, N. Banerjee, T. J. Cardinali, J. J. Yang, W. Wu, X. Li, W. M. Tong, D. B. Strukov, G. S. Snider, G. Medeiros-Ribeiro, and R. S. William, "Memristor—CMOS hybrid integrated circuits for reconfigurable logic," *Nano Lett.*, vol. 9, no. 10, pp. 3640–3645, 2009.

[5] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.

[6] B. Muthuswamy and P. P. Kokate, "Memristor-based chaotic circuits," *IETE Tech. Rev.*, vol. 26, no. 6, pp. 417–429, 2009.

[7] H. S. Majumdar, A. Bandyopadhyay, A. Bolognesi, and A. J. Pal, "Memory device applications of a conjugated polymer: Role of space charges," *J. Appl. Phys.*, vol. 91, no. 4, pp. 2433–2437, 2009.

[8] J. T. Moore and K. A. Campbell, "Memory device and methods of controlling resistance variation and resistance profile drift," U.S. Patent 6 930 909, Micron Technology, Inc., Boise, Idaho, ID, 2005.

[9] S. T. Hsu, "Temperature compensated RRAM circuit," U.S. Patent 6 868 025, Sharp Laboratories of America, Inc., Camas, WA, 2005.

[10] J. Straznicky, "Method and system for reading the resistance state of junctions in crossbar memory," U.S. Patent 7 340 356, Hewlett Packard, Palo Alto, CA, 2008.

[11] M.-J. Lee, Y. Park, B.-S. Kang, S.-E. Ahn, C. Lee, K. Kim, W. Xianyu, G. Stefanovich, J.-H. Lee, S.-J. Chung, Y.-H. Kim, C.-S. Lee, J.-B. Park, and I.-K. Yoo, "2-stack 1D-1R cross-point structure with oxide diodes as switch elements for high density resistance RAM applications," in *Proc. IEEE Int. Electron Devices Meeting*, Dec. 2007, pp. 771–774.

[12] D. Rinerson, C. J. Chevallier, S. W. Longcor, W. Kinney, E. R. Ward, and S. K. Hsia, "Re-writable memory with non-linear memory element," U.S. Patent 6 870 755, Unity Semiconductor Corporation, Sunnyvale, CA, 2005.

[13] Y. N. Joglekar and S. J. Wolf, "The elusive memristor: Properties of basic electrical circuits," *Eur. J. Phys.*, vol. 30, no. 4, pp. 661–675, Jul. 2009.

[14] G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, pp. 035204–035215, Jan. 2007.

[15] D. Rinerson, C. J. Chevallier, S. W. Longcor, E. R. Ward, W. Kinney, and S. K. Hsia, "Cross point memory array using multiple modes of operation," U.S. Patent 6 834 008, Unity Semiconductor Corporation, Sunnyvale, CA, 2004.

[16] G. Csaba and P. Lugli, "Read-out design rules for molecular crossbar architectures," *IEEE Trans. Nanotechnol.*, vol. 8, no. 3, pp. 369–374, May 2009.

[17] Y. Ho, G. M. Huang, and P. Li, "Nonvolatile memristor memory: Device characteristics and design implications," in *Proc. Int. Conf. Computer-Aided Design*, 2009, pp. 485–490.

[18] D. Niu, Y. Chen, and Y. Xie, "Low-power dual-element memristor based memory design," in *Proc. Int. Symp. Low Power Electron. Design*, 2010, pp. 25–30.

[19] W. Yi, F. Perner, M. S. Qureshi, H. Abdalla, M. D. Pickett, J. J. Yang, M.-X. M. Zhang, G. Medeiros-Ribeiro, and R. S. Williams, "Feedback write scheme for memristive switching devices," *Appl. Phys. A: Mater. Sci. Process.*, vol. 102, no. 4, pp. 973–982, 2011.

[20] L. M. Grupp, A. M. Caulfield, J. Coburn, E. Yaakobi, S. Swanson, P. Siegel,, and J. K. Wolf, "Characterizing flash memory: Anomalies, observations, and applications," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 24–33.

Idongesit E. Ebong (M'05) the B.S. and M.S. degrees in 2006 from Carnegie Mellon University, Pittsburgh, PA, both in electrical and computer engineering. He is currently working toward the Ph.D. degree in electrical engineering at the University of Michigan, Ann Arbor.

His research interest includes digital/analog integrated circuit design, focused primarily on new devices and low power applications.

Pinaki Mazumder (F'99) received the Ph.D. degree from the University of Illinois at Urbana-Champaign, Urbana-Champaign, in 1988.

Currently, he is a Professor with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He was the lead Program Director of the Emerging Models and Technologies Program at the US National Science Foundation, and worked for 6 years in industrial R&D centers that included AT&T Bell Laboratories, where in 1985, he started the CONES Project—the first C modeling-based very large scale integration (VLSI) synthesis tool. His research interests include current problems in Nanoscale CMOS VLSI design, CAD tools, and circuit designs for emerging technologies including Quantum MOS and resonant tunneling devices, semiconductor memory systems, and physical synthesis of VLSI chips.