Arithmetic Encoding for Memristive Multi-Bit Storage

Ravi Patel and Eby G. Friedman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, New York 14627
{rapatel,friedman}@ece.rochester.edu

Abstract— **Memristive memories have received significant interest for application to on-chip storage. A multi-bit memristive memory circuit architecture based on arithmetic coding is presented in this paper. Both read and write circuits are presented which encode information into the memristive data cells. The proposed circuits provide fine control of the resistance within the memristor. The continuous resistance characteristic of memristive devices is exploited to provide additional storage by utilizing compression techniques. This approach yields an increase in overall bit density for a memristor based data array as compared to a standard multi-bit cell array.**

I. Introduction

The discovery of a physical memristive device has prompted renewed interest in the field of memristive circuits. Memristors have been considered for a number of possible applications, ranging from programmable analog circuits to neuromorphic networks and solid-state memories. A memristive digital memory architecture is proposed herein utilizing the unique analog properties of these devices to compress digital information within a data array.

The proposed circuit leverages *a priori* knowledge of a bit sequence for storage. Through use of a compression algorithm with supporting circuitry, the circuit yields the potential to store significantly more bits per cell than a standard multi-bit approach. This approach is realized through a memristor driven sensing scheme and an adaptive write circuit that assign a resistance value to a memristive device with fine grain control.

In Section II, background on memristive devices and the proposed compression procedure is described. In Section III, the circuit architecture is reviewed. A description of the data modeling approach for memristive compression is presented in Section IV. A discussion of the simulation-based experimental results is presented in Section V. The paper is concluded in Section VI.

II. Background

The following section provides background information describing the characteristics of memristive devices as well as the specific features that enable an encoding based approach. A brief review of the applied coding scheme is also provided.

This research is supported in part by the National Science Foundation under Grant No. CCF-0829915, and grants from the New York State Office of Science, Technology and Academic Research to the Center for Advanced Technology in Electronic Imaging Systems, and by grants from Cisco Systems, Qualcomm Corporation and Samsung Electronics

1) Overview of memristors: Memristive devices can be described as non-volatile resistor-like devices whose conductance is modulated by an applied bias. Since memristors retain a written state when the voltage bias is removed, these devices are useful for low power storage applications. A large segment of memristive devices operate on the principle of dopant transport through the crystal lattice of a nanomaterial. In the well noted $\mathrm{TiO_2}$ memristor developed by Hewlett Packard in 2008 [1], the dopants are oxygen vacancies introduced into the material during fabrication.

The key feature of this specific type of memristor stems from the continuous "resistance" characteristic. The instantaneous resistance of a memristor [1] is

$$R = x \cdot R_{on} + (1 - x)R_{off}, \tag{1}$$

$$\frac{dx}{dt} = \frac{\mu_v Ron}{D^2} i(t) f(x), \tag{2}$$

where $i(t)$ is the applied current, μ_v is the drift velocity of dopants in the lattice, $x \cdot D$ represents the effective distance that the vacancies migrate within the device, and R_{on} and R_{off} represent, respectively, the minimum and maximum resistance of a memristor.

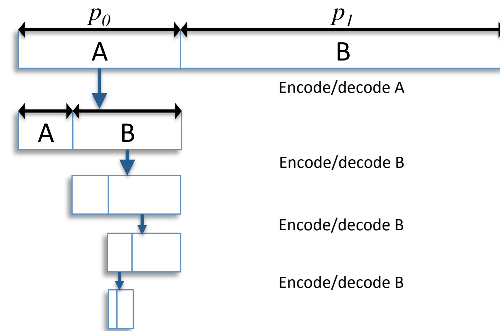


Fig. 1. Encoding process for two symbol alphabet and sequence $S_e = ABBA$. The initial interval is divided into sections corresponding to each symbol. The section size is governed by the probability of the symbol. Encoding S_e requires selecting the initial section that corresponds to A, subdividing this section according to the specified probabilities, selecting the subsection associated with C, and continuing the process until all symbols in S_e are encoded

2) Overview of arithmetic coding: Arithmetic coding is a long standing method for compressing data [2]. The procedure relies on assessing the probability of certain values within a data stream to create an encoding model that favors more

978-1-4673-2658-2/12/$31.00 ©2012 IEEE

frequent values. A string of bits is represented by a single compressed value. The continuum of potential compressed values can be encoded into a memristor because of the inherent continuous resistance characteristic of the device, improving the storage density of a memristor over standard multilevel approaches. The encoding process relies on mapping an uncompressed sequence to a fractional value within the interval $[0,1)$ which is related to a particular resistance within the resistive range of a memristive device. A probability model of a sequence informs the coding mechanism which encodes the data to a target resistance value.

Arithmetic coding uses a finite, non-empty set of elements A, designated as an *alphabet*. Each element $\{a_0, a_1, ..., a_k\}$ in the set, known as a *symbol*, represents a possible value within the data sequence being compressed A *sequence* is a series $S = s_n$ such that $\{s_n \epsilon A, \forall s_n \epsilon S\}$; this series represents an uncompressed data stream of symbols from the defined alphabet. A model $P = \{p_0, ...p_k\}$ associates each a_k in the alphabet with a probability p_k where $\sum_{i=0}^{k} p_i = 1$.

For example, consider the arbitrary sequence $S_e = ABBA$ for $A_e = \{A, B\}$, and the probability model $P_e = \{\frac{1}{4}, \frac{3}{4}\}$. This probabilistic model is defined according to the frequency of symbols within the sequence. The interval $[0,1)$ is divided into subintervals, each corresponding to a symbol in A_e. The length of each interval is equal to the probability associated with the corresponding symbol, as shown in Figure 1. The first detected symbol is A; the interval $[0, \frac{1}{4})$ represents the first symbol in the sequence. Any value within the interval is sufficient to encode the first bit of the sequence. To encode the second symbol, the interval $[0, \frac{1}{4})$ is again divided according to the probability model (see Figure 1). For the next symbol in the sequence (B), the interval $[\frac{1}{16}, \frac{1}{4})$ is selected which represents the top $\frac{1}{4}$ of the previous interval. A value within this interval encodes the first two symbols of the sequence. The process continues until all symbols in the sequence are encoded into a single value. The final interval for this example sequence is $[\frac{28}{256}, \frac{37}{256})$. Intuitively, the final interval is unique to the sequence S_e as other symbols would lead to different intermediate intervals. Selecting the value $\frac{32.5}{256}$ is, therefore, sufficient to encode the entire sequence.

The decoding process begins with the selected value ($\frac{32.5}{256}$) and the starting interval $[0,1)$. In a manner similar to the encoding process, the interval is partitioned according to the probability model. The selected value lies in the region of the interval corresponding to the symbol A. From this information, the first symbol in the sequence is decoded as A. Continuing to the second symbol, the interval $[0, \frac{1}{4})$ is selected and partitioned. The selected value occurs in the top $\frac{3}{4}$ of the interval $[0, \frac{1}{4})$ and corresponds to the symbol B, permitting the second symbol to be decoded. The process continues until the full sequence is retrieved. Through this process, a full data sequence can be reduced to a single fractional value without any loss of information.

These fractional values and the corresponding intervals are mapped to either a voltage or current by biasing a memristive device. The precise mapping mechanism is described in the following sections.

III. Memristive multi-bit encoding

The goal of memristive compression storage is to map a binary sequence to a fractional value using two symbol arithmetic encoding and store the value within a memristive data cell. A continuous set of encoded fractional values is mapped to the continuous resistance characteristic of a memristive device. The design of these circuits is predicated on two basic memristive building blocks, a resistive divider with adjustable memristors, and a memristor data cell containing the stored data. Circuits to both write and read a memristive data cell within the proposed encoding scheme are described in the following section.

A. Decoding and read circuitry

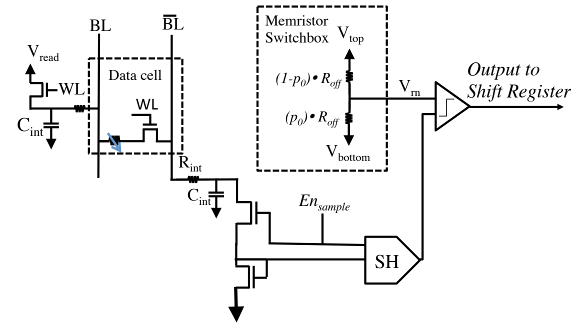


Fig. 2. Circuitry for reading an encoded value from a memristive data cell. Each read operation begins by selecting the cell in the data array which is compared against a reference voltage. The comparison is stored in a shift register at the end of each interim read operation. Depending upon the result of the comparison, either V_{top} or V_{bottom} is set to V_{rn}

The process of reading and decoding a data cell proceeds in a manner consistent with the compression process, as illustrated by the circuit shown in Figure 2. V_{bottom} and V_{top} begin with, respectively, the initial interval of V_{min} and V_{max}; these voltages correspond to the arithmetic coding interval $[0,1)$. Electrically, these levels are the maximum and minimum voltage biases that correspond, respectively, to the memristor states, R_{off} and R_{on}. The memristor values correspond to a probability model $P = \{p_0, 1-p_0\}$ for a two symbol alphabet $A = \{0,1\}$, where each memristor assumes the resistance value $p_k R_{off}$ for the two encoded symbols. When a read occurs, the voltage bias applied to the memristive data cell is set below the memristor threshold voltage. The current generated by this circuit is mirrored to a comparator. Within the first interval, a voltage divider generates the initial comparison voltage V_{rn}, where n represents the symbol being decoded (see Figure 2). The result of the comparison operation is stored in a shift register. Following this operation, if the result is logic 1, V_{bottom} is set to V_{rn}, otherwise V_{top} is set to V_{rn}. Setting V_{top} and V_{bottom} in this manner is the same procedure through which an arithmetic coding interval is selected for a given sequence. The voltage divider, with

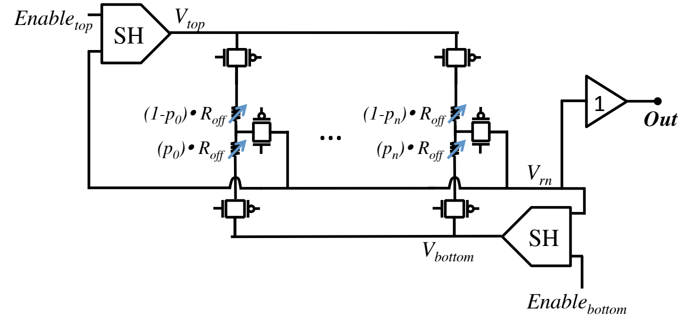
resistances set according to the probability of each symbol, generates the appropriate comparison threshold. This process continues until a maximum number of bits has been decoded. The initial voltage of the biased cell is stored within the sample and hold circuit, shown in Figure 2, to prevent writing to the memristor during an on-going read operation. The total number of bits stored per memristor is limited by the minimum distinguishable voltage on the output comparator. This limit is specified at design time and assumes that external decoding mechanisms detect when a specific output vector generates more bits than the noise level allows, and truncates the bit vector to the width.

Voltage divider switchbox: To properly modulate V_{top} and V_{bottom}, a circuit is required to both generate V_{rn} and to store intermediate values during the decoding process. This objective is accomplished by the voltage switchbox shown in Figure 3(a). Each sample and hold circuit drives a single pair of resistors. The resistance values of each pair correspond to the probabilities associated with a particular bitstream (*e.g.,* $p_0 = 0.1, 0.2...$). During the decoding process, V_{max} and V_{min} are applied, respectively, to V_{top} or V_{bottom}. The pair of resistors that correspond to the selected branch is switched on; the voltage division across the resistors gives rise to the threshold voltage V_{rn}. If the readout voltage is greater than the threshold voltage, the bottom sample and hold circuit is switched, otherwise the top sample and hold circuit is triggered. The sample and hold circuit stores the current value of V_{rn}. Switching the sample and hold circuit generates a new value for either V_{top} or V_{bottom}, producing the next threshold voltage V_{rn}. This process continues until the stored sequence is decoded.

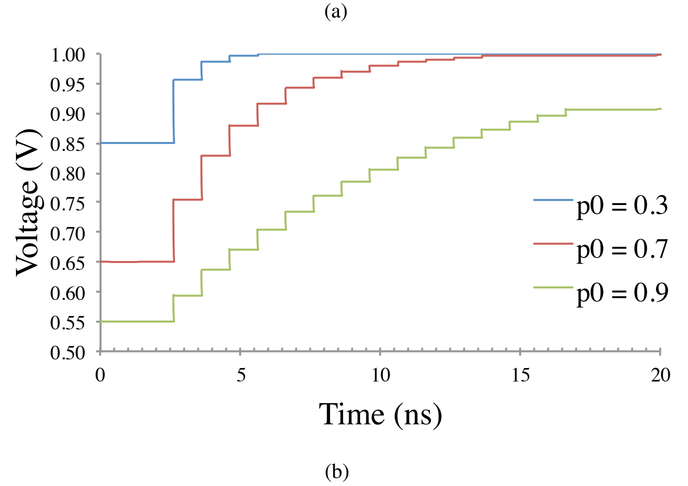
Operation of the circuit is illustrated in Figure 3(b). This graph depicts the voltage divider switchbox for an input bitstream containing only ones. A larger probability (p_0) indicates that ones are more prevalent in the input bitstream then zeros. Storing this specific sequence as a voltage level is more effective when the circuit is configured to a probability of 0.9 than the other two cases, resulting in a larger detectable difference between voltage levels. A larger detectable voltage level illustrates the process in which arithmetic encoding can be used to improve the storage density of a memristive device as compared to a traditional multi-bit approach.

B. Encoding and write circuitry

A variable-length data sequence is encoded into a single memristor by the circuity shown in Figure 4. The write operation occurs in three steps. First, the data being written, transmitted to the array in a pre-encoded state, creates a reference voltage using the switchbox. Afterwards, the wordline of the selected cell is biased high. Once En_0 and En_1 are switched on, the reference voltage is compared to a voltage generated at the output of the array. Given the bidirectional nature of memristive devices, this initial comparison, carried out by the write direction comparator shown in Figure 4, determines which direction to apply the bias for the write operation. This process establishes whether an increase or



(a)



(b)

Fig. 3. Voltage divider switchbox. (a) Circuitry for generating threshold voltages for both encoding and decoding circuitry. V_{bottom} and V_{top} are initially set to, respectively, V_{min} and V_{max}. If $Enable_{top}$ is set high, the sample and hold corresponding to V_{top} is set to the threshold voltage V_{rn}; the same is true for $Enable_{top}$. (b) Switchbox output for a bitstream of ones across different disparity levels.

decrease of the initial device resistance achieves the target value.

After this initial read, the second stage applies a voltage to the selected cell by raising the voltage on En_2. Applying a voltage to the memristor device for a prolonged period changes the device resistance. The write termination comparator continuously compares the two voltages indicated in Figure 4. The End signal is pulled low once the memristive device has been correctly written to the target resistance. Drift in the resistance, which occurs at the termination of a write operation, is a source of noise in the circuit. Note that the linear memristor model utilized in this analysis is known to be inaccurate [3]; however, the write procedure adaptively adjusts the target resistance to any write based on the electrical resistance of the device.

IV. Improvements In Bit Density

Encoding a fraction to a continuous memristor is only limited by the granularity at which the resistance can be changed, and the ability to distinguish values during read and write operations. For these operations, noise in the circuit as well as resistive drift governs the maximum number of bits

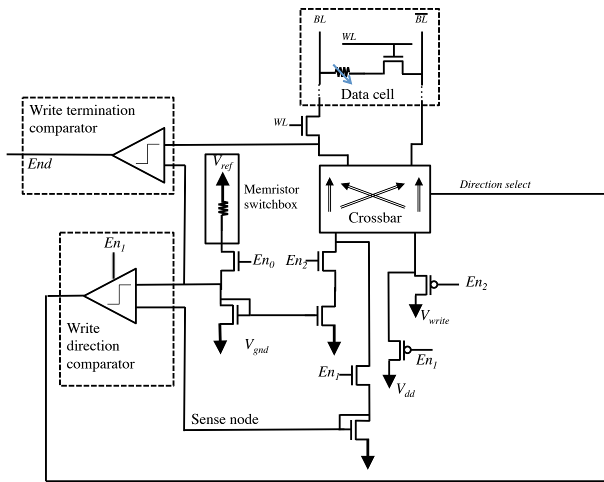


Fig. 4. The adaptive write circuit. An initial read of the selected data cell determines the direction required to write the device (En_1). This signal is relayed to the crossbar which selects the direction of the device. A fixed current is applied to both the reference switchbox and the data array (En_2). The write termination comparator indicates whether the state has been written. This event occurs when the voltage across the current mirror transistors is the same, fixing the equal currents and memristor resistance.

that can be stored within a memristor.

$$V_{min} \geq 2(V_n + V_{drift} + V_{SH}). \quad (3)$$

Equation (3) describes the minimum distinguishable voltage V_{min} within a memristive sensing operation. V_{drift} specifies the maximum voltage caused by resistive drift from the write operation, and V_{SH} represents the cumulative error from each of the sample and hold circuits. V_{drift} is due to the delayed termination of the write operation. For example, assume a change in resistance between 10 KΩ to 100 KΩ corresponds to an output voltage swing between 0 to 1 volts. If 25 mV of circuit noise is seen at the sensing circuitry and a 1 KΩ drift gives rise to a 25 mV error, the minimum distinguishable voltage would be 100 mV. Resistive drift and circuit noise are dependent on the circuit topology and resistive state of the device. The low resistance states drift more than the high resistance states due to the higher currents during the write procedure [4]. The sample and hold circuitry contribute three sources of error: the pedestal error associated with the sampling of a voltage level, the resistive drift caused by sampling during a read, and the droop rate of the hold state [5]. All three sources of error have a direct effect on the minimum distinguishable voltage.

For a simple two symbol alphabet, the disparity, a measure of the relative probability of symbols within an alphabet, is

$$disparity = abs((1 - p_0) - p_0) = abs(1 - 2p_0). \quad (4)$$

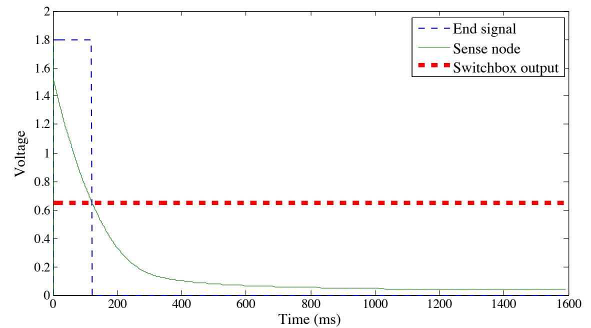
This metric describes the compression characteristics of a particular input bitstream. The storage capability of an array of memristive data cells can be characterized by this metric.

V. Experimental Evaluation

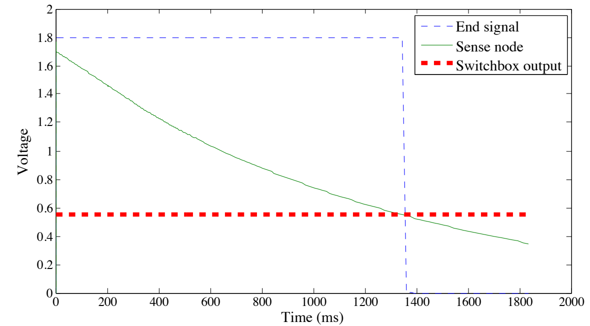
The proposed circuit architecture has been evaluated using a 1.8 volt, 180 nm CMOS technology. The memristor behavior is modeled by a linear VerilogA model [6]. This model corresponds to (1) and (2). Device parameters are from [1], and listed in Table I. For the purposes of this analysis, an ideal sample and hold circuit is assumed. The effects of non-idealities is assessed by the parameter V_{min}, as described in (3). The data stream is modeled as a random binary sequence. The arithmetic coding algorithm, applied to this sequence to determine the average improvement in bit density, is a function of the probability characteristics of the data stream and the tolerable noise. For simplicity, the probability is determined from the average occurrence of the symbols ($A = \{0, 1\}$) within the sequence.

TABLE I
Memristor Model Parameters [1]

u_v	1×10^{-14}	$m^2V^{-1}s^{-1}$
R_{off}	38	kΩ
R_{on}	100	kΩ
D	10	nm
V_{th}	1	V



(a)



(b)

Fig. 5. Adaptive write circuitry for target voltage levels (a) 650 mV, and (b) 550 mV. The *End* signal is pulled to ground when the device resistance has crossed the target threshold.

A. Circuit simulation

A simulation of the write circuitry is shown in Figure 5, which illustrates that a memristive device adaptively switches to the target voltage. As the memristor resistance changes, the voltage on the *Sense node* converges to the voltage specified by the switchbox. The *End* signal is pulled to ground when the memristive device surpasses the target voltage. A key limitation of this adaptive circuit is the wide range over which the device switching speed can vary. Switching from R_{on} to

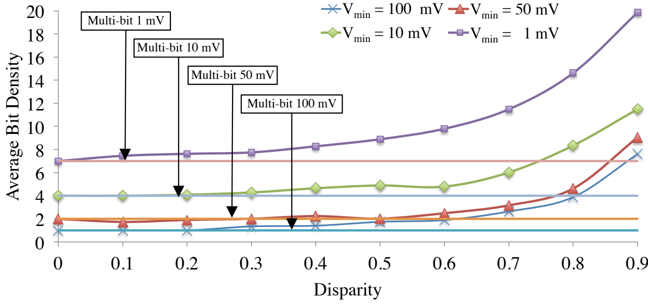


Fig. 6. Improvement in bit density versus disparity for increasing V_{min}

TABLE II
AVERAGE BIT DENSITY VS V_{min}

Disparity	100 mV	50 mV	10 mV	1 mV
0	1	2	4	7
0.1	1	1.7309	4.0015	7.4546
0.2	1	1.9029	4.0855	7.6279
0.3	1.344	2.0048	4.2803	7.7496
0.4	1.4055	2.2359	4.6454	8.2768
0.5	1.742	2.0182	4.8872	8.8869
0.6	1.8945	2.4858	4.7909	9.7958
0.7	2.6363	3.1652	6.0143	11.483
0.8	3.8793	4.6234	8.3456	14.622
0.9	7.6021	9.0552	11.498	19.867

the voltage level shown in Figure 5(a) requires approximately 100 ns; however, switching to the level shown in Figure 5(b) requires more than 1.3 s. The adaptive scheme has a one-to-one correspondence between a write bias voltage and a memristor state. In this adaptive scheme, higher resistance states correspond to lower write bias voltages. As a result, switching to a higher resistance state causes the switching process to require more time than if a full voltage bias is applied to the circuit.

The maximum voltage range delivered to the memristor varies between 500 mV (V_{min}) and 980 mV (V_{max}). This range considers the voltage drop across the access transistors and the adaptive current mirror (which is utilized during write operations).

The resistive drift of the device during this process is shown to be negligibly small. This small drift is due to the slow switching speed observed in the device, which is on the order of milliseconds. The total peak V_{drift} is observed to be 0.3 μV, comparable to the thermal noise generated by a memristor in the on state. Resistive drift is therefore neglected.

B. Bit density

The minimum noise level determines the storage density as a function of the data disparity. The bit density is illustrated in Figure 6 and listed in Table II. The case of no disparity models a traditional multi-bit approach, where the voltage range is divided equally by the minimum increment in observable voltage (V_{min}). For this comparison, the voltage drop across the access devices for a traditional multi-bit approach is assumed to be the same as the encoded approach.

An improvement in storage density over a traditional approach is seen for all cases, however, only a marginal improvement is noted for those data sets with a disparity less then 0.5. The average bit storage density can, however, be improved by a factor of 7.6 for high noise, high disparity data sets. The overall improvement in storage density is dependent on the relative frequency of the different sequences.

VI. CONCLUSIONS

A circuit architecture is presented which supports arithmetic encoding of data within memristive data cells. Novel read and write circuits are described that support fine grain control and detection of the memristor device resistance. The application of the encoding procedure exhibits storage density improvements of 7.6x for a specific data set. Future studies will determine the effect of the relative frequency of various data sets with differing disparity on the storage density.

REFERENCES

[1] D. Strukov, G. Snider, D. Stewart, and R. Williams, "The Missing Memristor Found," *Nature*, Vol. 453, No. 7191, pp. 80–83, May 2008.

[2] A. Said, "Introduction to Arithmetic Coding - Theory and Practice," Tech. Rep. HPL-2004-76, HP Laboratories, Palo Alto, CA, April 2004.

[3] J.J Yang, M. D. Pickett, X. 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] 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, pp. 074508, 2009.

[5] D. Johns and K.W. Martin, *Analog Integrated Circuit Design*, John Wiley & Sons, 1997.

[6] A. G. Radwan, M. A. Zidan, and K. N. Salama, "On the Mathematical Modeling of Memristors," *Proceedings of the IEEE International Conference on Microelectronics*, pp. 284–287, December 2010.