A Novel Scan-In Scheme for CMOS/ReRAM Programmable Logic Circuits

Md Musabbir Adnan
Sherif Amer
Garrett S. Rose

May, 2018

IEEE International Symposium on Circuits and Systems (ISCAS)

https://ieeexplore.ieee.org/document/8351185/

A Novel Scan-In Scheme for CMOS/ReRAM Programmable Logic Circuits

Md Musabbir Adnan, Sherif Amer and Garrett S. Rose
Department of Electrical Engineering & Computer Science
University of Tennessee Knoxville
Knoxville, Tennessee 37996
Email: {madnan, samer1, garose}@utk.edu

Abstract—**Resistive RAM (ReRAM) devices are low power, fast and reliable nanoelectronic memory devices, which have proven to be crucial in both neuromorphic hardware and memory design. Despite their utility, challenges remain in forming and programming these devices before they can be used in a system. This paper builds on a forming circuit, incorporating programming technique in the same circuit in addition to outlining a scan-in approach to programming both CMOS and ReRAM memory elements. The proposed scheme utilizes a single input pin to serially scan-in a bit sequence to program the memory elements, leveraging a digital control circuitry. The suggested protocol is applied to a neuromorphic system to configure synaptic properties to implement a spiking neural network. Lastly, the need for reduction of forming voltage is highlighted with power dissipation data from Spectre simulation.**

I. Introduction

Memristors [1] have become ubiquitous in many VLSI systems. Their analog programmability and resistive state retention have made them useful in systems such as memories and neuromorphic circuits where those properties can be leveraged. ReRAM devices are Transition Metal Oxide (TMO) based memristors. While several other materials with different switching mechanisms have demonstrated memristive properties, ReRAMs have been predominantly used for their CMOS compatibility specially with Back-End-Of-Line portion of the process flow, small footprint, reliability and comparable understanding of their switching dynamics. ReRAM can be thought of a resistive switch that acquires resistance values between the maximum and minimum resistance states namely, High Resistance State (HRS) and Low Resistance State (LRS), respectively. The rate of change of the resistance is a function of the polarity and value of the applied voltage.

In many VLSI systems, the logic circuits need to be configured to perform specific applications. For example, in neuromorphic circuits, the synapses connecting the neurons has to be configured to account for the weight and delay [2]. This calls for programmability of both CMOS and ReRAM devices as the CMOS memory configures the delays, while memristors represent synaptic weights. The weights need to be programmed off-line wherein initial weights are uploaded to the circuit before on-line learning can overtake during runtime. Ideally, every CMOS memory or memristor can be accessed through a pin with some supporting circuitry and programmed to its corresponding state. However, this approach might be impractical in a typical neuromorphic circuit where several hundreds of memroy elements or memristors are required. This work proposes a novel approach for programming CMOS memory as well as memristors in any CMOS/ReRAM based reconfigurable systems. The proposed approach optimizes the area utilizing a scan-in protocol for programming CMOS memory and memristive devices in a typical system wherein the programming bits are uploaded serially. As a case study, the proposed scheme is applied to a neuromorphic architecture named Memristive Dynamic Adaptive Neural Network Array (mrDANNA) [3].

The remainder of the paper is organized as follows: section II discusses the methods to form and program ReRAM devices. Section III presents our scan-in scheme for programming CMOS memory and ReRAM devices. Section IV provides an example design with the proposed scheme. Section V evaluates the performance in terms of power consumption of the circuitry. Section VI concludes the paper.

II. Background

The switching mechanism of ReRAM devices is governed by the formation and rupture of conductive filaments. A ReRAM sample in its pristine state is typically an insulator that does not exhibit any resistive switching properties until electroforming is executed. Electroforming (also known as forming) is a one time process that initiates defects (usually oxygen vacancies) in the oxide which later contribute to resistive switching. While threshold voltages are typically low or, at least, within the range of operation voltages of state-of-the-art CMOS technology, forming voltages, on the other hand, are typically high. In [4], an in-field forming circuit was proposed to overcome this challenge. The proposed circuit isolates the peripheral circuitry which cannot tolerate high forming voltages and executes forming in-field.

Programming ReRAM devices also presents a challenge, especially, in neuromorphic systems where analog resistance tuning is required. In [5], a memristor programming circuit was proposed which is capable of analog state tuning of each device. However, the area overhead of such circuitry is significant. In addition, the stochasticity of filament growth may render this approach infeasible. Hence, in this work, binary programming is adopted while tuning the resistance state is done during learning for its inherent resilience to such variability.

III. Programming ReRAM Devices

A. Forming and Programming Circuit

While ReRAM devices are advantageous as aforementioned, the requirement of a one time forming process adds complexity to ReRAM-based circuits. Forming voltages are typically higher than nominal voltages used during cycling (i.e. normal Set/Reset operations) which may not be tolerable for regular FET devices. For example, in this work, regular FET devices operate at $1.2V$ while the forming voltage is $2.1V$ based on HfO_2 ReRAM devices manufactured at SUNY Polytechnic Institute [4]. Hence, we use DGXFET devices provided by the IBM PDK 65nm CMOS 10LPe, which operates at $3.3V$. The forming circuit implemented herein (Fig. 1) is a modified version of the circuit in [4] which uses the model developed in [6] to account for forming voltage. It also includes level shifter circuit (not shown in the figure) to convert the control signals from $1.2V$ to $3.3V$ as needed.

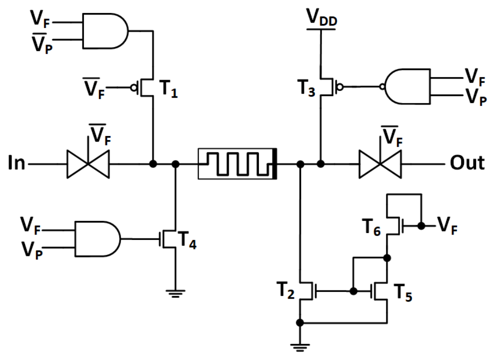

Fig. 1: Forming and Programming circuit

Another important step in the specified system is programming the memristive devices to initial weights. While analog state tuning is ideal, the significant circuit overhead required in such approach may deem it impractical. Hence, we follow in this work a different strategy where memristors are programmed to either LRS or HRS and count on online learning to fine tune the weights during runtime. During SET operation (Switching from HRS to LRS) a current compliance may be required to ensure a self limiting process [7] is activated and avoid a downward drift in LRS. In order to achieve compact design, the programming step was integrated in the forming phase such that the current compliance represented by transistors $T2$, $T5$ and $T6$ in Fig. 1 can be shared during both forming and SET operations. The circuit operation can be best described by the following truth table in table I. When $V_F = 0$, the whole circuit is deactivated and the series transmission gates connect the memristors to the learning circuit. When $V_F = 1, V_P = 0$, forming and SET operations are executed where the current compliance mechanism is activated for the aforementioned reasons. When $V_F = 1, V_P = 1$, RESET is executed.

TABLE I: Forming and programming scheme

V_F	V_P	F
0	X	$learning$
1	0	$Forming/SET$
1	1	$RESET$

B. Programming Logic

In this approach, both D Flip-Flops and ReRAM devices are programmed using one single input pin. The input pin called *Bit_Stream* provides a serial input with 'K' bits for the flip-flops and 'N' bits for ReRAM programming. It is connected to the input of a 1-to-2 De-multiplexer. The selector of the De-Mux (*DONE_FF* signal in Fig. 2) chooses either the D Flip-Flops or the ReRAM devices to program. This *DONE_FF* signal indicates the end of D Flip-Flop programming and is generated by a counter that keeps track of the timing of the programming. It is a digital counter, that counts up to 'K' so that the 'K' number of D Flip-Flops can be programmed. After 'K' bits are loaded into the D Flip-Flops, the counter changes the select signal of De-Mux from '0' to '1'. This change in level is converted into a pulse that resets the count value to '0'. The pulse is generated by a Moore type finite state machine and is applied for one clock period. Now, the *Bit_Stream* is connected to ReRAM programming path. A D Flip-Flop synchronizes the programming bits of the ReRAM devices to the counter values. The output of the D Flip-Flop connects to a 1-to-N De-multiplexer via an AND gate. The other input of the AND gate is *DONE_FF* signal which ensures that ReRAM programming does not begin until after the Flip-Flop programming is finished. The 1-to-N De-Mux is connected to the 'N' ReRAM devices which are selected by the counter. The input *Bit_Stream* is passed sequentially to the 16 ReRAM devices. A '1' in the bit stream programs the ReRAM from HRS to LRS. A '0' keeps the ReRAM device at HRS.

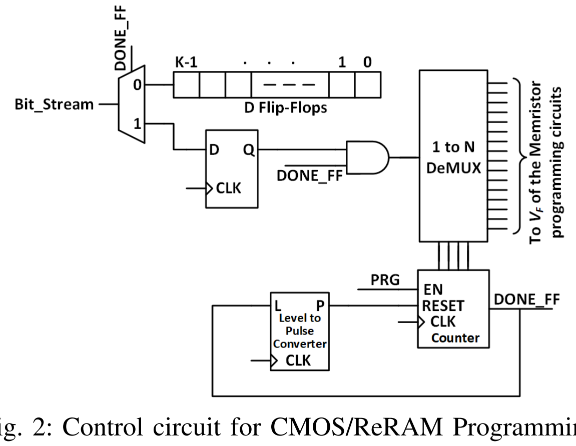

Fig. 2: Control circuit for CMOS/ReRAM Programming

The programming follows a protocol that starts with forming all the memristors simultaneously. A global forming pulse is applied to all the memristor programming circuits. V_F is asserted $3.3V$ while V_P is $0V$. Hence the forming path is active and current flows through $T1$, memristor and $T2$

transistor. After the memristors are formed, they are at LRS. Then a RESET pulse is applied to all the memristors. V_F is kept at $3.3V$ and V_P is also asserted $3.3V$. The LRS to HRS programming path becomes activated as current flows through $T3$, memristor and $T4$. Now all the memristors go to HRS so that they can be selectively programmed back to LRS using the control circuit of Fig. 2. We chose to selectively program from HRS to LRS because the transition from HRS to LRS is much faster than the transition from LRS to HRS [8]. If the memristors were formed first and then selectively programmed from LRS to HRS, longer pulse would have been needed, resulting in reduction of clock frequency.

After the initialization protocol is completed, all the memristors are in HRS. Now the scan-in procedure starts and a pre-configured bit sequence is applied at the *Bit_Stream* pin. For this example simulation result, 3 bit D Flip-Flop and 4 ReRAM devices are programmed. The timing waveforms of the signals are shown in Fig. 3. At time $t = 2.5\mu s$, the programming begins. At first the 3 D Flip-Flops are programmed, then the memristors M0, M1, M2, M3 are sequentially programmed according to the bit sequence provided. The bit sequence was '1101011' which programs the D Flip-Flops to '011' (reverse order as the first input bit goes to the last Flip-Flop of the chain) and the memristors M0, M1, M2, M3 are programmed to LRS, HRS, LRS and LRS respectively. The memristances are plotted in Fig. 4, which shows the memristance values during both initialization and selective programming phase.

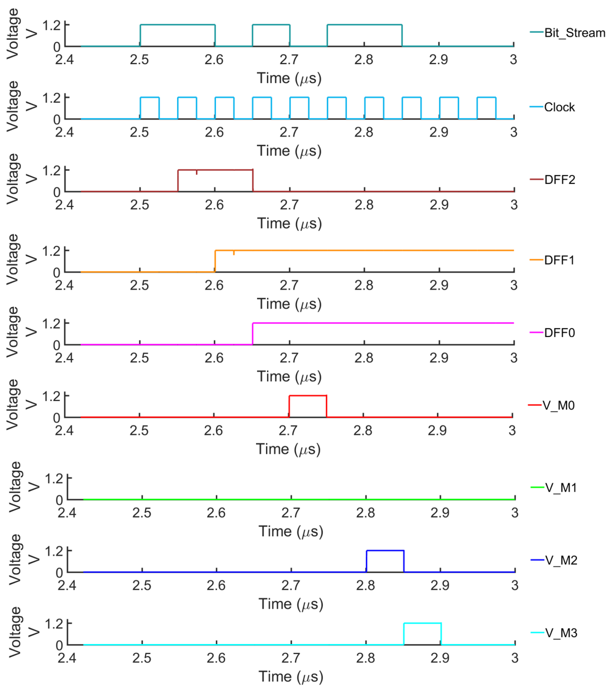

Fig. 3: Timing diagram to program the CMOS memory and selectively program ReRAM devices after initialization

IV. MrDANNA Core Programming

In [3], a memristive Neuromorphic system called mrDANNA is described which is based on Dynamic Adaptive Neural Network Array (DANNA) [2]. The total mrDANNA system is divided into cores which act as processing units. Each core houses n number of synapses and a neuron, as shown in Fig. 5. Each synapse has two properties associated with it, namely $delay$ and $weight$. Synaptic delay is realized by using D Flip-Flops as delay unit inside the synaptic buffer. As for the synaptic weights, mrDANNA system uses analog ReRAM devices. 2 ReRAMs connected with opposite polarity comprises the synapse, which can give both positive and negative weights. Synaptic buffer drives positive current through one memristor (M_p) and negative current through the other (M_n). The effective current depends on the memristance of the ReRAM devices. Hence, the conductivity or the weight of the synapse is given by the following equation.

$$G_{eff} = \frac{1}{M_p} - \frac{1}{M_n} \quad (1)$$

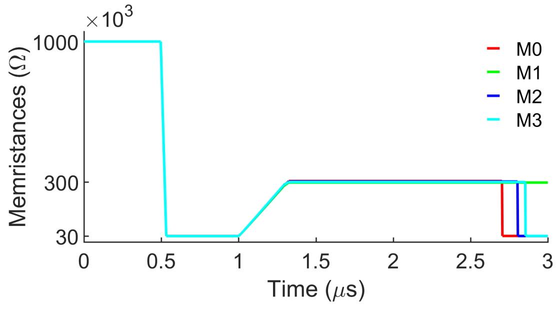

Fig. 4: Forming and programming of 4 ReRAM devices. M0, M2, M3 are programmed to LRS while M1 is kept at HRS

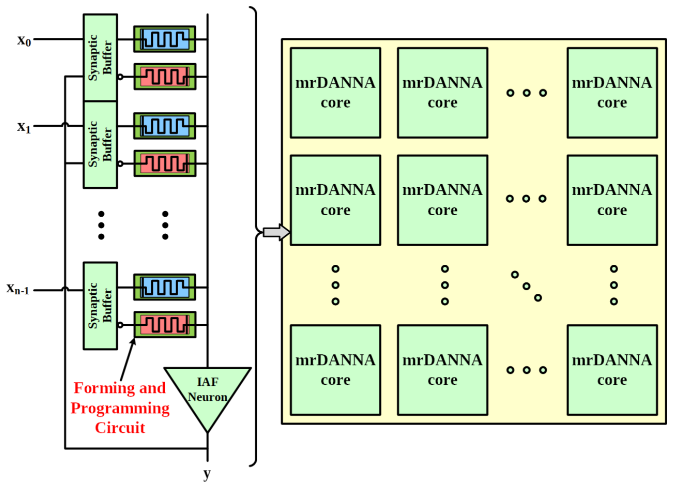

Fig. 5: Memristive DANNA architecture with forming and programming circuits

MrDANNA system implements a Spiking Neural Network (SNN) featuring evolutionary optimization algorithm for off-line learning [9] and Spike Timing Dependent Plasticity for on-line learning [10]. The off-line algorithm is used to generate an initial network with synaptic delays and weights, from which the on-line learning rule can nudge the synaptic weights incrementally to implement desired tasks. It may be noted that the synapse delay is not changed during on-line learning. For simplicity of programming ReRAM devices, the off-line algorithm initializes the network with 3 synaptic weights, namely maximum, minimum and zero weights. Maximum weight is realized by programming M_p to LRS and M_n to HRS. Conversely, minimum weight is realized by programming M_p to HRS and M_n to LRS. Weight of zero is given by programming both M_p and M_n to LRS.

In this example, each mrDANNA core consists of 8 synapses. Each synapse can have maximum delay of 7 cycles, which is configured by the delay block. It uses 3 flip-flops to store the configuration bits. So, in total 24 flip-flops are needed to be programmed along with 16 memristors. A 6 bit counter synchronizes the control circuit, scanning-in 24 bits for the Flip-Flops. Then the control circuit resets the counter and activates the ReRAM programming path to program the 16 ReRAMs according to the scan-in serial input. The network programmed thus with 3 sets of weight reaches a classification accuracy of 95.7% for Iris dataset with online learning [11].

V. Performance Analysis

The performance of the proposed scheme is analyzed by measuring the power dissipation of the circuits used. For power measurement, the programming of 3 D Flip-Flops and 4 ReRAM devices were considered, as described in section III-B. The control circuit of Fig. 2 is a purely digital block and consumes only $1.46\mu W$ with a V_{DD} of $1.2V$. In contrast, the forming and programming circuit uses a V_{DD} of $3.3V$ as the forming voltage is typically higher. So the power consumption of the forming and programming circuit is high and depends on HRS and LRS values of the ReRAM devices. The circuit was simulated with different HRS/LRS values of ReRAMs found in the literature [12]. The resulting power consumption data is shown in Fig. 6. With the increase of HRS/LRS values, the power consumption drops significantly, as the current through the ReRAM device drops lower.

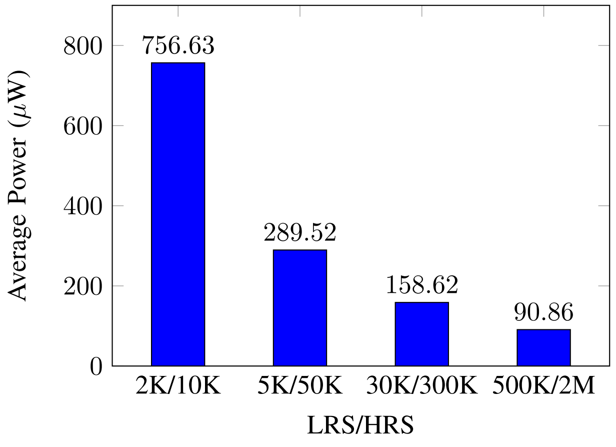

Fig. 6: Avg. power for forming and programming for different LRS-HRS ranges of ReRAM devices with V_{DD} = $3.3V$.

The high power consumption during the forming phase is mainly due to the high forming voltage that has to be applied for the ReRAM devices to form. If the forming voltages can be reduced, then the supply voltage, V_{DD} can be less than $3.3V$. To quantify the effects of reducing forming voltage and in-turn supply voltage V_{DD}, forming voltage parameter in the ReRAM device model was varied to measure power consumption for different V_{DD}. The results in Fig. 7 show an order of magnitude reduction in power consumption if the supply voltage is reduced from $3.3V$ to $1.2V$.

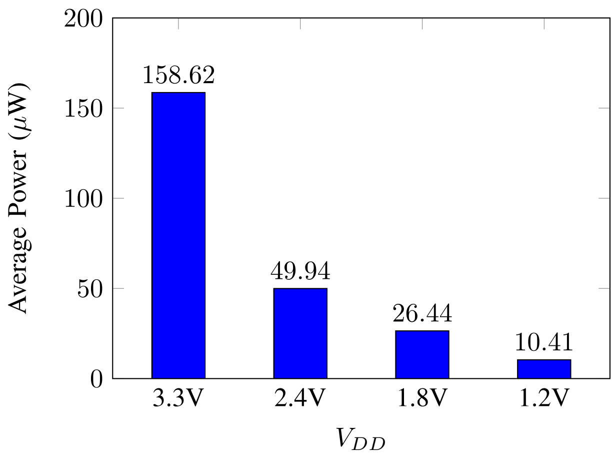

Fig. 7: Avg. power for forming and programming for different supply voltages with LRS/HRS of 30K/300K.

VI. Conclusion

This work proposes a method to reuse the forming circuit of ReRAM devices to selectively and sequentially program them in conjunction with the CMOS memory elements. The programming protocol defined here reduces the pin count, scanning-in the data bits serially. The proposed scheme drops analog memristor programming, in favor of digital approach, programming only to HRS or LRS. It relies on the learning algorithm of the neuromorphic system to achieve precise analog memristance value to reach the solution. A study on the effect of device parameters such as LRS/HRS value and forming voltage is also presented suggesting the need for a push towards higher device memristance and lower forming voltages or forming-free devices.

Acknowledgment

This material is based in part upon research sponsored by Air Force Research Laboratory under agreement number FA8750-16-0065. Portions of this work were performed at Oak Ridge National Laboratory with funding and support from the Laboratory Directed Research and Development Program. This manuscript has been authored in part by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The authors would like to thank Sagarvarma Sayyaparaju and Gangotree Chakma of University of Tennessee Knoxville for interesting and useful discussions on this topic.

References

[1] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[2] M. E. Dean, C. D. Schuman, and J. D. Birdwell, "Dynamic adaptive neural network array," in *International Conference on Unconventional Computation and Natural Computation*. Springer, 2014, pp. 129–141.

[3] G. Chakma, S. Sayyaparaju, R. Weiss, and G. S. Rose, "A mixed-signal approach to memristive neuromorphic system design," in *60th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Boston, MA, August 2017.

[4] S. Amer, G. S. Rose, K. Beckmann, and N. C. Cady, "Design techniques for in-field memristor forming circuits," in *Proceedings of IEEE International Midwest Symposium on Circuits and Systems MWSCAS*.

[5] R. Berdan, T. Prodromakis, and C. Toumazou, "High precision analogue memristor state tuning," *Electronics letters*, vol. 48, no. 18, pp. 1105–1107, 2012.

[6] S. Amer, S. Sayyaparaju, G. S. Rose, K. Beckmann, and N. C. Cady, "A practical hafnium-oxide memristor model suitable for circuit design and simulation," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

[7] D. Ielmini, "Modeling the universal set/reset characteristics of bipolar rram by field-and temperature-driven filament growth," *IEEE Transactions on Electron Devices*, vol. 58, no. 12, pp. 4309–4317, 2011.

[8] K. Beckmann, J. Holt, H. Manem, J. Van Nostrand, and N. C. Cady, "Nanoscale hafnium oxide rram devices exhibit pulse dependent behavior and multi-level resistance capability," *MRS Advances*, pp. 1–6, 2016.

[9] C. D. Schuman, J. S. Plank, A. Disney, and J. Reynolds, "An evolutionary optimization framework for neural networks and neuromorphic architectures," in *International Joint Conference on Neural Networks*. Vancouver: IEEE, July 2016.

[10] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *Nanoscale Architectures, 2008. NANOARCH 2008. IEEE International Symposium on*. IEEE, 2008, pp. 85–92.

[11] A. Wyer, M. M. Adnan, B. W. Ku, S. K. Lim, C. D. Schuman, R. C. Pooser, and G. S. Rose, "Evaluating online learning in memristive neuromorphic circuits," in *Neuromorphic Computing Symposium: Architectures, Models, and Applications (NCAMA)*, Knoxville, TN, July 2017.

[12] M. Uddin, M. B. Majumder, and G. S. Rose, "Robustness analysis of a memristive crossbar puf against modeling attacks," *IEEE Transactions on Nanotechnology*, vol. 16, no. 3, pp. 396–405, 2017.