Memristive Devices for Stochastic Computing

Siddharth Gaba, Phil Knag, Zhengya Zhang, Wei Lu
Electrical Engineering and Computer Science
University of Michigan – Ann Arbor
Ann Arbor, Michigan, USA
wluee@eecs.umich.edu

***Abstract* — We show resistive switching effects in memristive devices exhibit significant stochasticity. When the switching is dominated by a single filament, the switching time is fully random and shows a broad distribution. However, the switching distribution can be predicted and responds well to controlled changes in the programming conditions. The native stochastic characteristic can be used to generate random bit streams with predictable biases that can lead to efficient and error-tolerant computing.**

Keywords—memristor, stochastic, filament, bit stream.

I. INTRODUCTION

Memristive devices are a leading contender for future non-volatile memory technology due to their scalability [1], high endurance [2] and ultra-low power consumption [3,4]. Relatively simple fabrication, multi-bit capability [5] and ease of 3D stacking [6,7] allow significant reduction in the cost per bit metric. However, a significant roadblock to the transition of memristors to main stream manufacturing and successful commercialization of these devices is the large variations in the switching parameters. These devices inherently suffer from significant spatiotemporal variations [8, 9] – the switching parameters vary from device to device and even cycle to cycle in the same device. Device to device variations (spatial variations) arise due to issues like line edge roughness and film thickness irregularity and these can be corrected through improved processes and variation aware design. The more challenging problem with memristor devices is the significant temporal randomness experienced during normal operation. This randomness is due to the stochastic nature of filament formation in resistance switching that leads to the memristive effects. The randomness has been systematically studied experimentally to reveal that this supposed randomness, in fact, can be well characterized and controlled. When a single filament growth dominates, the randomness in the switching processes is shown to follow a Poissonian distribution which lends a native, fully stochastic characteristic to these devices. This native characteristic can be used for novel stochastic computing schemes. Instead of trying to use these inherently non-deterministic devices for deterministic applications (like digital memory storage), the underlying non-deterministic fabric can be used in a hybrid CMOS-memristor architecture where the randomness enables more efficient computing.

II. MEMRISTOR AS AN INHERENT STOCHASTIC DEVICE

A. Stochastic Switching Behaviour

The temporal variation in switching parameters in conventional resistive switching devices has been studied systematically recently [10]. While a metal/insulator /semiconductor structure comprising of silver/amorphous silicon/p doped poly-silicon was used for the study, the results can be generalized to other memristive systems (e.g. devices based on oxygen vacancy migration [11]) as well. The binary switching devices (Fig 1a, inset) were fabricated in a crossbar form (Fig 1c, inset) and individual devices were engineered for a relatively high threshold voltage of ca. 5V (Fig 1a, inset). The abrupt change from OFF state to ON state is explained by formation of a dominant Ag filament within the amorphous silicon matrix [12,13]. To reveal the underlying randomness in the switching parameters, bias voltages lower than the threshold voltage were applied to the silver top electrode while keeping the poly silicon bottom electrode grounded. Current through the device was continuously monitored until a sharp jump in the current (indicating the OFF to ON transition) was observed (Fig 1b, inset), and the wait time leading to the switching event was recorded. The device was then reset to OFF state and the measurement was repeated one hundred times for each bias voltage. The resultant wait times were systematically analyzed to study the temporal variation in a single device.

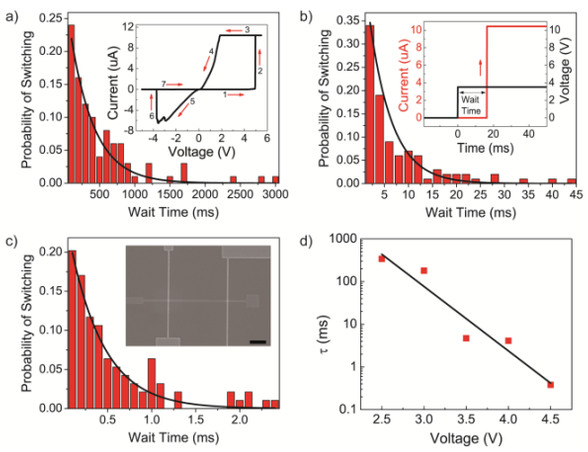

Fig. 1. Random distribution of the wait time prior to switching. (a–c) Distributions of wait times for applied voltages of 2.5 V (a), 3.5 V (b) and 4.5 V (c). Solid lines: fitting to the Poisson distribution (1) using τ as the only fitting parameter. τ = 340 ms, 4.7 ms and 0.38 ms for (a)–(c), respectively. Insets: (a) DC switching curve, (b) example of a wait time measurement and

(c) scanning electron micrograph of a typical device (scale bar: 2.5 μm). (d) Dependence of τ on the programming voltage. Solid squares were obtained from fitting of the wait time distributions while the solid line is an exponential fit.. Reproduced from [10].

As shown in Fig 1, the wait time is not a fixed quantity and varies broadly from cycle to cycle in a single device. The nanoscale filament formation associated with the OFF to ON transition is thermodynamically driven and involves multiple physical processes like oxidation, ion transport and oxidation, while one of the processes will play a dominate role. Given all these processes are thermally activated, thermal activation over the dominant energy barrier is probabilistic in nature if only a dominant filament is involved[12]. Theoretically, the wait time is expected to follow a Poisson distribution such that the probability of a switching event occurring within Δt at a time t is given by

$$P(t) = \frac{\Delta t}{\tau} e^{-\frac{t}{\tau}} \quad (1)$$

where τ is the characteristic wait time [12].

Excellent fit to (1) shown in Fig 1 for each applied voltage (2.5V (Fig 1a), 3.5V (Fig 1b) and 4.5V (Fig 1c)) verifies the hypothesis of thermal activation over a dominant energy barrier and, thus, the inherent stochastic nature of these devices. The switching time of the device is then governed by the characteristic wait time. Fig 1d shows that the characteristic wait time decreases exponentially as a function of the applied voltage, and drops by almost three orders of magnitude when the applied voltage is increased by 2V. This is also coherent with the dominant filament model since both the oxidation and transport processes are field dependent and the effective activation barrier is reduced by the applied bias voltage resulting in exponential speed up in switching time [12,14, 15].

B. *Predictable Randomness*

An important significance of (1) is that the switching probability *C(t)* can be calculated by simply integrating the Poisson distribution from (1).

$$C(t) = 1 - e^{\frac{-t}{\tau}} \quad (2)$$

For an applied voltage of 2.5V the predicted curve from (2) is shown as solid line (Fig 2a) while the experimental values from Fig 1a are shown as squares. Excellent matching between the predicted and obtained data indicates the validity of the stochastic hypothesis. The ability to predict the switching probability was further tested by single pulse measurements with different pulse widths (300ms and 1000ms) at a fixed voltage amplitude of 2.5V. After each pulse, the device state was verified – ON or OFF – and the device was reset to original OFF state to prevent the result of one programming pulse affecting the results from subsequent tests. For trials with 300ms pulses, the device switched to the ON state twenty times out of fifty attempts(Fig. 2c) while this number increased to thirty eight (out of fifty attempts) for attempts with 1000ms pulses (Fig 2d). These numbers are very close to those predicted from (2) (Fig 2b). If the result of each trial can be treated as a 1 or a 0 depending on the device current measured at the end of the voltage pulse, then a stochastic bit stream can be generated in this fashion. For example, if the current through the device is larger (smaller) than 0.1uA, the resultant bit is treated as a 1(0). Thus for fifty trials, we can obtain a fifty bit long string where the ratio of number of 1s to the total string length (or bias) is completely determined by the voltage and time duration of the pulse used to program the device. For example, for 2.5V/300ms pulses a stream of bits (or a bit stream) with a bias of 0.4 was generated while the bias changed to 0.76 for longer pulse width of 1000ms. The ability to generate bit streams and predict the number of switching events or bias by altering the pulse width has unique implications in fields like stochastic computing.

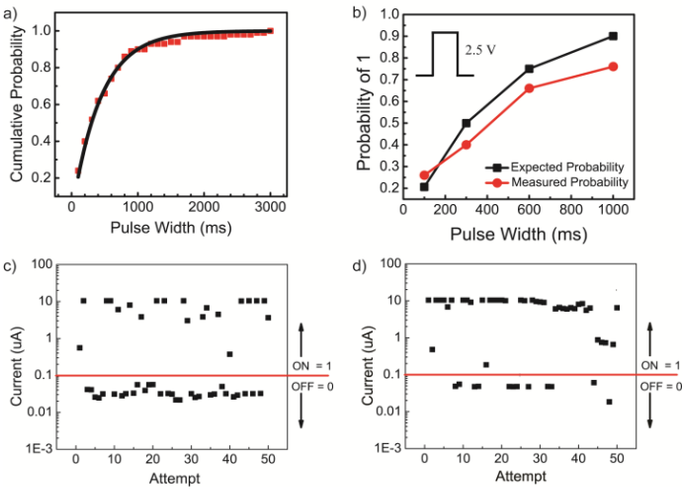

Fig. 2. Probability of switching within a single pulse. (a) Pulse width dependence. The solid line shows values predicted from (2), squares were obtained by measuring the cumulative probability obtained from Fig. 1a. (b) Expected and measured probability using a single 2.5 V pulse. (c–d) Device current measured after repeated application of a single 2.5 V, 300 ms (c) and 1000 ms (d) pulse. The device switched 20 times (out of 50 attempts) for the 300ms pulse and 38 times (out of 50 attempts) for the 1000ms pulse expriment. Reproduced from [10].

III. STOCHASTIC COMPUTING AND MEMRISTORS

A. *Stochastic Operations*

Stochastic computing was first proposed in the 1960s [16] as a low cost computing paradigm that used an interesting numerical representation involving streams of bits. Unlike a traditional binary encoded number format, stochastic computing represents numbers by the probability that the bit in the given bit stream is equal to one. For example, a bit stream A is made up of 100 bits, with forty 1's and sixty 0's randomly permuted together represents the number $a = 0.4$ because there is a 0.4 probability that a randomly chosen bit is 1. Unlike base-2 binary numbers, there are multiple representations of the same stochastic number with the exception of $a = 0$ and $a = 1$ (given that the bit streams are of the same length). One of stochastic computing's primary benefits is its potential to cheaply implement some forms of computation. For example, the multiplication of two numbers a and b can be implemented very efficiently in this stochastic number system with an AND gate (Fig. 3). To understand how this works, consider bit streams a and b with corresponding

probabilities P_a and P_b. The probability of a bit at a given position in both stream *a* and *b* are both 1 is equal to $P_a \times P_b$.

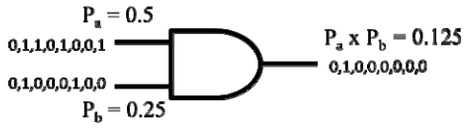

Fig. 3. Stochastic multiplication using a logic AND gate.

The stochastic bit stream representation is error resilient since all the bits in the bit stream are LSBs. Therefore, a few bit flips in any given stochastic number will not significantly alter any given result. Furthermore, since the order of the bits in a stochastic bit stream has no meaning, unlike binary encoded numbers, bit level parallelization becomes straightforward. For example, a stochastic multiplier can be parallelized by running multiple AND gates in parallel.

B. *Randomization Challenge*

Stochastic computing relies on random bit generation. In stochastic arithmetic, the operand bit streams are assumed to be independent. Any correlation between operand bit streams degrades the accuracy of the calculation. For example, multiplying two identical bit streams representing *a* gives P_a, not $P_a \times P_a$. The randomization is done by stochastic number generators (SNGs) [17] as shown in Fig 4. Adding SNGs in systems significantly increases overheads, sometimes as high as 80% of the total resource usage [18]. Further, SNGs need to be inserted along multiple intermediate stages to mitigate correlations introduced by reconvergent signals. The extensive deployment of SNGs easily surpasses core logic as the dominant cost in stochastic computing. This added overhead of SNGs has been a major roadblock in the widespread adoption of stochastic computing.

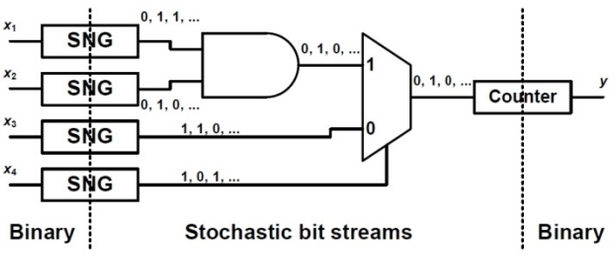

Fig. 4. Stochastic implementation of logic function $y= x_1x_2x_3 + x_3(1- x_4)$

Implementing SNG in CMOS is expensive because CMOS inherently lacks true randomness, and the randomness needed by SNGs must be created with psuedo-random number generators like linear-feedback shift registers (LFSRs). However, given that memristors have inherent randomness that can be captured very easily without complicated circuitry, memristors are a potentially ideal implementation of SNGs because of their low cost and high integration density. The problem of generating independent random bit streams is solved by simply programming multiple memristive devices with identical voltage pulses. Using memristors to produce uncorrelated bit streams with nearly identical biases has been experimentally shown in [10]. Here four different devices were used to obtain twenty bit long bit streams (Fig 5). While the bias for a given pulse height and width can be predicted from (2), the exact position of the 1s and 0s in the bitstream is completely random. Thus, the devices are used to produce streams of bits which are uncorrelated but the total number of 1s to the string length in each case is almost the same (0.6 in this case). In other words, four different independent bitstreams representing the number 0.6 have been generated without resorting to expensive SNGs.

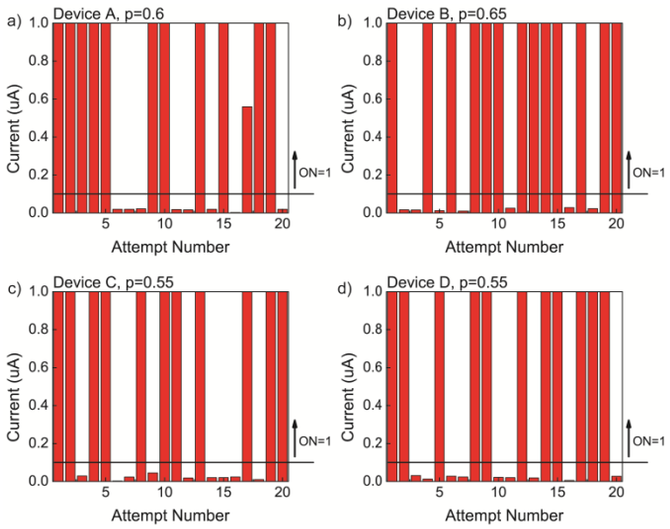

Fig. 5. Generation of non-correlated bit streams. Different devices (A-D) when programmed with identical bias conditions give non correlated bit streams but with very similar bias (~0.6). Reproduced from [10].

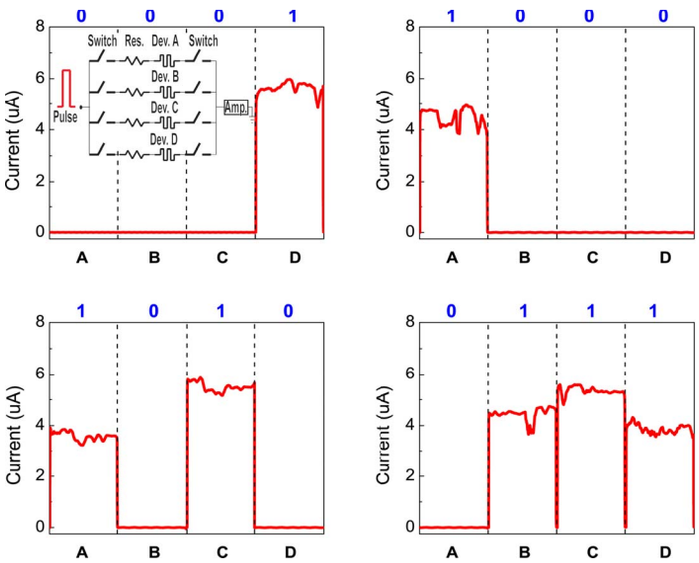

Fig. 6. Stochastic programming of a memristive device array using a simple circuit (inset) . Four representative combinations {0001} (a) {1000} (b) {1010} (c) and {0111} (d) are shown here out of a total of $2^4 = 16$ combinations. The different combinations were obtained in the same array under identical pulses. The devices were programmed in parallel using a single 2.75 V, 1000 ms pulse and their states were measured using 2 V, 100 ms pulses individually after the programming pulse. The array was reset after each measurement. Reproduced from [10].

The above methodology of programming multiple devices sequentially produces each bit at a different time instance. In other words, the bit streams thus produced are "stochastic in time". For more efficient parallel computing applications, the bit streams may also be produced in parallel using memristive devices connected in an array fashion. For example, using an array of four parallel connected memristive devices, four bits were produced each time the array was programmed (Fig 6). Each time the array is programmed a 4 bit long stream is produced where each bit corresponds to whether or not the corresponding device transitioned to the SET state in a random fashion. Such bit streams, termed stochastic in space, are useful in applications requiring bit level parallelism.

A lot of scientific research recently has been focused on using stochastic computing methodologies in areas like image processing [18] and artificial neural networks [19]. All these applications fit the stochastic operating paradigm where errors can be tolerated. Further, in applications where the required accuracy changes with time, stochastic processors allow trading off accuracy for energy consumption by simply using longer or shorter bit streams. By using a longer bit stream, accuracy of the system increases at the cost of increased time and energy required to produce these bit streams. Compared to SNGs in binary systems, memristors allow more efficient production of bit streams and thus are suitable for integration into stochastic systems for current and future compute-intensive but error tolerant probabilistic applications.

IV. CONCLUSION

In summary, we show memristive devices have an inherent randomness built into them and need no extra overhead or circuitry to maintain and ensure this randomness. The wait time for a single device at identical programming conditions is not fixed, but varies randomly and has a broad distribution. On the other hand, tuning the applied voltage and pulse width can control the overall probability of switching. This unique property of "predictable randomness" can be used to produce bit streams, in both time and space domains, with a known bias. Such bit streams produced using memristors can be efficiently used in stochastic computing systems in place of resource intensive stochastic number generators; thereby reducing overall system complexity and cost.

ACKNOWLEDGMENT

This work was supported in part by the AFOSR through MURI grant FA9550-12-1-0038 and by the National Science Foundation (NSF) through grant CCF-1217972. This work used the Lurie Nanofabrication Facility at the University of Michigan, a member of the National Nanotechnology Infrastructure Network (NNIN) funded by NSF.

REFERENCES

[1] B. Govoreanu, G. S. Kar, Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini, I. P. Radu, L. Goux, S. Clima, R. Degraeve, N. Jossart, O. Richard, T. Vandeweyer, K. Seo, P. Hendrickx, G. Pourtois, H. Bender, L. Altimime, D. J. Wouters, J. A. Kittl, M. Jurczak, B.- Leuven, and K. U. Leuven, "10x10nm 2 Hf / HfO x Crossbar Resistive RAM with Excellent Performance , Reliability and Low-Energy Operation," Electron Devices Meeting (IEDM), IEEE International, pp.31-6 , 2011.

[2] M.-J. Lee, C. B. Lee, D. Lee, S. R. Lee, M. Chang, J. H. Hur, Y.-B. Kim, C.-J. Kim, D. H. Seo, S. Seo, U.-I. Chung, I.-K. Yoo, and K. Kim, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta2O(5-x)/TaO(2-x) bilayer structures.," Nat. Materials, vol. 10, no. 8, pp. 625–630, 2011.

[3] A. Chin, C. H. Cheng, Y. C. Chiu, Z. W. Zheng, and Ming Liu, "Ultra-Low Switching Power RRAM Using Hopping Conduction Mechanism Albert Chin," E. C. S. Transactions, vol. 50, no. 4, pp. 3–8, 2012.

[4] J. P. Strachan, A. C. Torrezan, G. Medeiros-Ribeiro, and R. S. Williams, "Measuring the switching dynamics and energy efficiency of tantalum oxide memristors.," Nanotechnology, vol. 22, no. 50, p. 505402, 2011.

[5] J. Park, K. P. Biju, S. Jung, W. Lee, J. Lee, S. Kim, S. Park, J. Shin, and H. Hwang, "Multibit Operation of TiO x -Based ReRAM by Schottky Barrier Height Engineering," Electron Device Letters, IEEE, vol. 32, no. 4, pp. 476–478, 2011.

[6] I. G. Baek, D. C. Kim, M. J. Lee, H. Kim, E. K. Yim, M. S. Lee, J. E. Lee, S. E. Ahn, S. Seo, J. H. Lee, J. C. Park, Y. K. Cha, S. O. Park, H. S. Kim, I. K. Yoo, U. Chung, J. T. Moon, and B. I. Ryu, "Multi-layer Cross-point Binary Oxide Resistive Memory (OxRRAM) for Post-NAND Storage Application," Electron Devices Meeting (IEDM), IEEE International, IEDM Technical Digest, pp. 750-753, 2005.

[7] I. G. Baek, C. J. Park, H. Ju, D. J. Seong, H. S. Ahn, J. H. Kim, M. K. Yang, S. H. Song, E. M. Kim, S. O. Park, C. H. Park, C. W. Song, G. T. Jeong, S. Choi, H. K. Kang, and C. Chung, "Realization of vertical resistive memory (VRRAM) using cost effective 3D process", .Electron Devices Meeting (IEDM), IEEE International, pp.31-8 , 2011.

[8] S. Yu, S. Member, X. Guan, and H. P. Wong, "On the Switching Parameter Variation of Metal Oxide RRAM — Part II□: Model Corroboration and Device Design Strategy," IEEE Transactions on Electron Devices, , vol. 59, no. 4, pp. 1183–1188, 2012.

[9] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," Int. Reliab. Phys. Symp., p. MY.7.1–MY.7.4, 2011.

[10] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications.," Nanoscale, vol. 5, no. 13, pp. 5872–5878, 2013.

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

[12] S. H. Jo, K.-H. Kim, and W. Lu, "Programmable resistance switching in nanoscale two-terminal devices.," Nano Lett., vol. 9, no. 1, pp. 496–500, 2009.

[13] Y. Dong, G. Yu, M. C. McAlpine, W. Lu, and C. M. Lieber, "Si/a-Si core/shell nanowires as nonvolatile crossbar switches.," Nano Lett., vol. 8, no. 2, pp. 386–391, 2008.

[14] D. B. Strukov and R. S. Williams, "Exponential ionic drift: fast switching and low volatility of thin-film memristors," Appl. Phys. A, vol. 94, no. 3, pp. 515–519, 2008.

[15] H. Schroeder, V. V. Zhirnov, R. K. Cavin, and R. Waser, "Voltage-time dilemma of pure electronic mechanisms in resistive switching memory cells," J. Appl. Phys., vol. 107, no. 5, p. 054517, 2010.

[16] B. R. Gaines, "Stochastic computing," Spring Joint Computer Conf., pp. 149–156, 1967.

[17] X. Li, W. Qian, M. D. Riedel, K. Bazargan, and D. J. Lilja, "A reconfigurable stochastic architecture for highly reliable computing," Proc. 19th ACM Gt. Lakes Symp. VLSI - GLSVLSI , p. 315, 2009.

[18] W. Qian, S. Member, and X. Li, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," vol. 60, no. 1, pp. 93–105, 2011.

[19] B. D. Brown and H. C. Card, "Stochastic neural computation. I. Computational elements," IEEE Trans. Comput., vol. 50, no. 9, pp. 891–905, 2001.