Usages of Memristors

Bailey Denzer
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
denz0045@morris.umn.edu

ABSTRACT

Recently, advances in traditional computer memory technologies have begun to decelerate as *CMOS*(complementary metal-oxide-semiconductor) currently used to construct integrated circuits) based computing is reaching physical barriers to scaling. This has resulted in less than spectacular improvements to performance, energy efficiency, and density. Some researchers and companies are starting to see alternative computing architectures as viable replacements to provide continued incremental improvements to computing performance. One such technology is the memristor developed by a research team headed by R. Stanley Williams in 2008. These memristors are being developed in order to implement the crossbar array that has the potential to replace standard transistor based CPU architectures and many different memory formats as the memristor has the potential to be a *universal memory* (a data storage device that is suitable for all types of memory storage). It meets the requirements of a universal memory as it theoretically has read/write times faster than DRAM, densities greater than DRAM, non-volatility, long life span, and it is also suitable for performing logic.

Keywords

memristor, crossbar array, computation-in-memory, 1T1R, non-volatile, memristive

This first paragraph will need to be reworded as it is a copypaste of the abstract

1. INTRODUCTION

Recently, advances in traditional computer memory technologies have begun to decelerate as *CMOS*(complementary metal-oxide-semiconductor) based computing is reaching physical barriers to scaling. This has resulted in less than spectacular improvements to performance, energy efficiency, and density. Some researchers and companies are starting to see alternative computing architectures as viable alternatives to continue the incremental improvements to computing performance. One such technology that has recently gained interest is the memristor developed by a Hewlett Packard research team headed by R. Stanley Williams in 2008. These memristors are being developed in order to implement the crossbar array, which has the potential to replace traditional transistor based CPU architectures and many different memory formats as the memristor has the potential to be a *universal memory* (a data storage device that is suitable for all types of memory storage). It meets the requirements of a universal memory as it theoretically has read/write times faster than DRAM, densities greater than DRAM, non-volatility, long life span, and it is also suitable for performing logic.

Needs more work Section two of this paper will first give background information on the memristor which will go over its properties as well as attempt to cover the history of the memristor as it is somewhat of a long story. A short introduction to the crossbar array will also be provided in the background. In section 3 one of the applications of the memristor that makes use of the crossbar array will be covered along with some results of simulations using the architecture compared to conventional computing. Another application called 1T1R that uses memristors in a more conventional approach without the introduction of the crossbar array will also be described in section 4. The 1T1R section will also be provided with information on the read and write properties of the memristor. The paper will then be concluded with section 5.

2. BACKGROUND INFORMATION

2.1 The Memristor

Left this section alone for the most part as I was getting conflicting suggestions on what to do here

The *memristor* is a *non-volatile* memory which retains information without power supply a state of resistance depending on the history of applied voltage. It is constructed by placing a layer of highly resistive Titanium Dioxide, TiO_2, and a conductive layer of TiO_2 which has had some amount of oxygen atoms removed that is represented as TiO_{2-x}. These two layers are placed between two electrodes. These electrodes would most likely be made of platinum and would be part of the crossbar array which will be covered later. The state of the memristor is altered by passing a voltage through it.

A simplified explanation of how this works is as follows. When a positive voltage is passed through the electrode on

UMM CSci Senior Seminar Conference, December 2015 Morris, MN.

the TiO_{2-x} side of the memristor it will cause vacancies to be pushed down into the TiO_2. This results in the TiO_{2-x} increasing in thickness while the TiO_2 shortens. This increase in width of the conductive region and decrease in width of the resistive region. It also significantly reduces the overall resistance as resistance of a material is determined by both its resistive properties and overall length. The overall width of a memristor can constructed to be only a few nano meters or less, so it doesn't take much of a change in size of the resistive layer to have a large impact on the resistance. This results in the memristor having a $R_{\text{On}}/R_{\text{Off}}$ ratio > 1000, where R_{Off} and R_{On} are the resistances of the memristor that correspond to 0 or 1, respectively. This also happens fairly quickly, giving memristors switching times of nano seconds. In order to read the memristor without disturbing its state, it is simple enough to use alternating current since alternating the current will result in no net movement of the oxygen vacancies.

When reading up on the memristor it is hard not to get confused as it has a somewhat complicated history. The memristor was originally hypothesized as the fourth missing circuit element by Chua in 1971, with the others being the resistor, capacitor, and inductor [3]. The theoretical memristor specifies a linkage between magnetic flux and charge. It was not until recently in 2008 that claims of the physical realization of the memristor were made by R. Stanley Williams's research team at Hewlett Packard [7]. There has been debate over whether the 2008 memristor really is the theoretical memristor. The 2008 "memristor" was not the first device created to exhibit memristive properties, however these devices were never claimed to be memristors. In fact, the memristive properties of TiO_2 were known as far back as 1966, but the research was not cited by William's et al. in 2008 [1] [7]. Leon Chua replied to these criticisms in 2011 by stating that all memristive memories can be classified as memristors [2]. In 2015 Vongher claimed that the current "memristor" is more than likely another implementation of existing memristive memory types, as the conceptual memristor is not possible without magnetic induction [8]. The HP memristor is likely another implementation of oxidation based RRAM (resistive RAM), which uses compounds with similar properties to the memristor to store information as resistance. In fact, many research papers use the terms memristor and RRAM interchangeably. This should not effect the results of the research though, as it would appear that the physical properties of the "memristor" are true to the explanation given above, and that the criticism is that the "memristor" is not equivalent to the original theoretical memristor. For simplicity and to avoid confusion I will continue to refer to the HP device as a memristor as that is what the research I am covering calls it.

2.2 Crossbar Array

When reading research on memristors it is rare to not find mention of the crossbar array. The crossbar array is simply a collection of switches arranged in a matrix pattern. It was originally invented by HP in 2001. From then on HP researchers were searching for a switching material with a high enough Off/On ratio to work with it. Implementing a nano-scale crossbar array was the motivation behind the development of the memristor by R. Stanley Williams's team [7]. The interest in the crossbar array is due in part to its impressive scaling capabilities (down to 5 nm) and its simplicity allows it to be cheaply manufactured [5]. More information will be provided on the crossbar array in the CIM section. (Still thinking about where everything should go, might move all crossbar array things up here).

Table 1. Comparison of data storage technologies. (Data drawn from public sources and HP internal research)

	Memristor	PCM	STT-RAM	DRAM	Flash	HD
Chip area per bit (F^2)	4	8–16	14–64	6–8	4–8	n/a
Energy per bit (pJ)[2]	0.1–3	2–100	0.1–1	2–4	10^{4}–10^{5}	10^{6}–10^{7}
Read time (ns)	<10	20–70	10–30	10–50	25,000	5–8x10^{6}
Write time (ns)	20–30	50–500	13–95	10–50	200,000	5–8x10^{6}
Retention	>10 years	<10 years	Weeks	<Second	~10 years	~10 years
Endurance (cycles)	~10^{12}	10^{7}–10^{8}	10^{15}	>10^{17}	10^{3}–10^{6}	10^{15} ?
3D capability	Yes	No	No	No	Yes	n/a

Table 1: [4] Comparison of different memories. From left to right: PCM is phase change memory, STT-RAM is MRAM (magnetic RAM), DRAM (dynamic RAM) is currently used for main memory, NAND or flash memory used in SSD's, and HD. At the moment everything other than DRAM, NAND, and HD are currently still under development

2.3 Advantages of Memristors

Figure 1 provides a chart of some of the advantages memristors have over other memory storage types [4]. The source is from an article by HP, but comparing the results to similar RRAM tables these results seem to be accurate. They have significant advantages in density, read/write time, data retention, cycles (endurance), energy consumption. gCrossbar arrays allow them to be densely stacked into a three dimensional block with the potential to store petabytes of memory within a cubic centimeter. Not shown in the chart, memristors have also drawn particular interest as they can potentially be cheaply manufactured using techniques for manufacturing traditional CMOS technology. When looking at the table it would also appear that DRAM is only slightly less energy efficient than memristors but this is not true. What is shown is the power to access a bit of DRAM once, but it does not account for the fact that DRAM must continually be refreshed, causing it to consume large amount of power relative to the other memories in the table.

3. CIM - COMPUTATION IN MEMORY

Memristors are not restricted to applications as memory storage. CIM (computation in memory) is being proposed by [5] as a replacement for traditional Von Neumann (VN) architecture for use in data-intensive applications. Figure 1 gives a simple visual for both VN and CIM architecture. With traditional architecture processing and data storage are performed in physically separate locations. This results in the need for large amounts of data transfer between the two. Von Neumann architecture has some draw backs that the CIM architecture can address. For data intensive computations, having large amounts of data being transferred back and forth between the processor and memory can result in the processors being unable to reach their maximum performance as the processor needs to wait for information transfer, and energy inefficiency as the data transfer can be

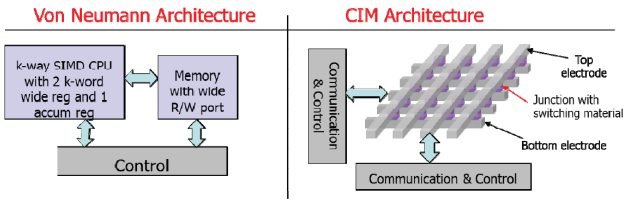

Figure 1: Von Neumann architecture on left, CIM on right.

costly. It is estimated that communication of data takes up 70 to 90% of energy consumption in VN architecture, with the rest being used for actual computation [5]. This is a result of the processor-memory bottleneck, often called the Von Neumann bottleneck. Within the crossbar array that CIM utilizes, the memristors can be dynamically chosen to be used for either gates or data storage. This allows large portions of the crossbar array to be set up so that data necessary to a computation can be physically stored next to the memristors that have been converted into gates. This removes the need to be constantly accessing things like SRAM caches which have high energy consumption, poor density (due to the fact that it takes between 4 to 10 transistors to represent a bit in SRAM), and information bottle-necking. The motivation behind realizing a CIM architecture is to eliminate the communication bottleneck, support massive parallelism, and reduce energy inefficiency.

A more succinct overview of the advantages of CIM is as follows:

- Tightly integrated (scalable to 5nm) computation-in-memory crossbar architecture supporting massive parallelism.
- Little to no power leakage. Current VN architectures are beginning to run into scaling problems due to power leakages from constantly accessing SRAM caches.
- Significant performance improvement at lower energy and area.

3.1 Basic Design

In CIM the computation and data storage are integrated in a crossbar array. Memristors in a crossbar array can used for data storage, switching, and logic which allows all three to be integrated into the same chip. This removes the need for DRAM based main memory and SRAM caches. Since the crossbar array has such high densities and the memristors themselves have little to no power leakage and read/write switching time, there would be enough room for these to be combined without needing to increase the size of the processor to match the total storage/processing power. The memristor based crossbar construction also uses materials and designs that CMOS manufacturing techniques can be applicable to. Outside of the crossbar array, the control and communication can be made using standard CMOS based technology, but it would require a significant redesign to be compatible with a CIM architecture. Because of this, the research paper from Hamdioui et al. is proposing CIM as a concept and does not appear to have a fully designed architecture at hand. Computations of this architecture are simulated based off of assumptions of how the architecture would behave.

Metric	Archit.	DNA Sequencing	10^6 additions
Energy-delay/ operations	Conv.	2.0210e-06	1.5043e-18
	CIM	2.3382e-09	9.2570e-21
Computing efficiency	Conv.	4.1097e+04	6.5226e+09
	CIM	3.7037e+07	3.9063e+12
Performance area	Conv.	5.7312e+09	5.1118e+09
	CIM	5.1118e+09	4.9164e+12

Table 2: Results of simulated data-sets. Unfortunately the source only used these as comparisons, and did not provide any actual units.

Case	p	q	p→q
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

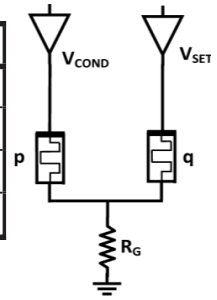

Figure 2: The imply function. The initial values of p and q are the input bits, with q being the output bit after applying the voltages V_{set} and V_{cond} [6]

3.2 Results on Large Data-sets

Table 2 displays the simulated results between conventional and CIM architecture from [5]. The DNA sequencing results were computed by performing a common solution for comparing two DNA sequences in which a sorted index of reference DNA is created in order to identify the locations of matches or mismatches in another sequence. In the particular case set up by Hamdioui et al. they are comparing 200 GB of DNA data to 3GB of healthy reference. The other example is for 10^6 parallel addition operations. The first row of the table represents the the energy-delay product per operations, the second row the computation efficiency defined as the number of operations per required energy, and the third row is the number of operations per area. These simulations were based off of a fairly large table of assumptions provided by Hamdioui et al. [5]. Some of the assumptions appear to be questionable through, as their assumption on the read time for a memristor is 200 ps which they cited from ITRS which is a fairly reputable source. However this read time does not match up with what other sources are reporting which is closer to 2 ns, roughly 10 time what is being assumed here. Because of this, it is likely that the results are more realistically one or two orders of magnitude in difference, rather than 3.

3.3 How Memristors Function as Logic Switches

An interesting thing about memristors is that they do not implement the same logic that conventional transistor based gates would. In order to perform logic in a memristor cross-

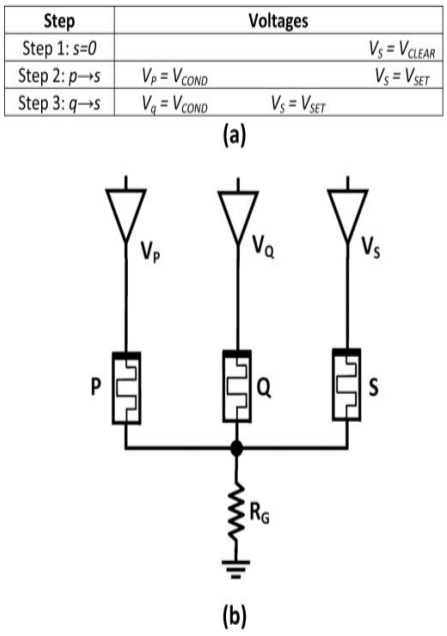

Figure 3: A nand gate using IMP logic [6]

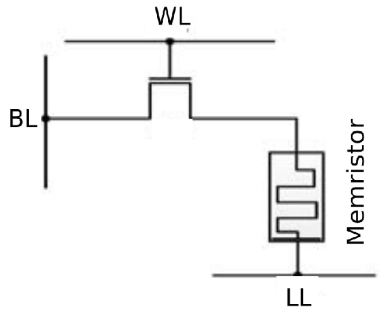

Figure 4: 1T1R cell: BL = Bitline, WL = Wordline

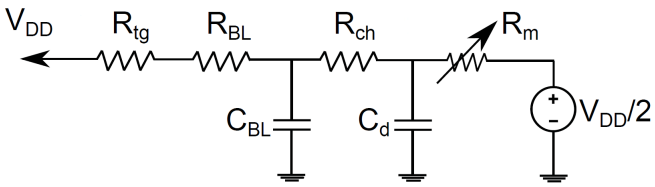

Figure 5: Circuit of a 1T1R cell during a write. V_{BL} is set to V_{DD} for writing 1, and 0 for writing 0.

bar array architecture, material implication logic (IMP) is used. A truth table of the imply function along with an imply logic gate is given in Figure 3.3. The imply funciton can be used to create more complex gates, allowing for crossbar arrays of memristors to perform any logic functions. An example of an imply based nand gate along with the steps that occur in its execution is shown in Figure 3.3. In either of the gates the memristors are connected to a resistor R_G where the resistance of R_G is between R_{Off} and R_{On}. What exactly happens during the imply gate with memristors P and Q with initial states p and q is as follows:

- Apply voltage V_{cond} to P and V_{set} to Q where ($|V_{\text{cond}}| < |V_{\text{set}}|$)
- If p = 1 (low resistance), the voltage on the shared terminal is approximately V_{cond} and the voltage on Q is $V_{\text{set}} - V_{\text{cond}}$, which is small enough that the state of Q is unchanged.
- If $p = q = 0$ (high resistance), then the resistance on Q is approximately V_{set}, and Q is switched to 1.
- If p = 0 and q = 1, the state of Q is also unchanged.

This section still needs a better explanation, and I still need to convert the graphs into pdf format

4. 1T1R

One of the main problems with the crossbar array is the sneak path problem. Since electricity always looks for a path with the least resistance, there is little preventing the previously described crossbar array from having what is called a sneak path. This results when an alternative path is made when reading/writing where instead of going through the desired memristor, the current may instead travel through different memristors. This is far from an unsolved problem as there are multiple ways of dealing with this, but one method in particular is the 1T1R cell as proposed by Zangeneh et al. [9]. It stands for one transistor, one resistor (in this case the resistor is a memristor) and has some parallels with DRAM cells, as they are made up of one transistor, one capacitor. A 1T1R equivalent cell during a write operation is shown in figure 4. Like DRAM, the proposed 1T1R architecture has a wordline to select a row of cells and a bitline to select columns. During a write a voltage V_{DD} is applied to the wordline, and a positive or negative voltage is applied across the memristor for writing 1 or 0 logic. This is done by either charging the bitline to V_{DD} (for logic 1) or discharging it to 0 V (for logic 0), and applying a voltage $\frac{V_{DD}}{2}$ at node LL.

This implementation would likely not be the best choice to use in the aformentioned CIM architecture. While it would still allow for data storage and computation within the same region, it would be less than ideal to be replacing transistors in a processor with a transistor/memristor hybrid.

4.1 Read/Write Properties of 1T1R

This subsection will provide the four main equations for the read and write models, regarding the time to read/write and the energy to read/write. Comparing the results of the models to the results to Table 1, they seem to agree with each other. Figure 5 provides a circuit diagram during a write procedure.

4.1.1 Write Time Model

The equation to model the time necessary to write is:

$$T_w = \frac{L^2(1+\beta)}{2\mu_v V_A}$$

L is the thickness of the memristor, β is the ratio of R_{Off} and R_{On}, V_A is the magnitude of the applied voltage, and μ_v is a constant for the mobility of the oxygen vacancy. Accord-

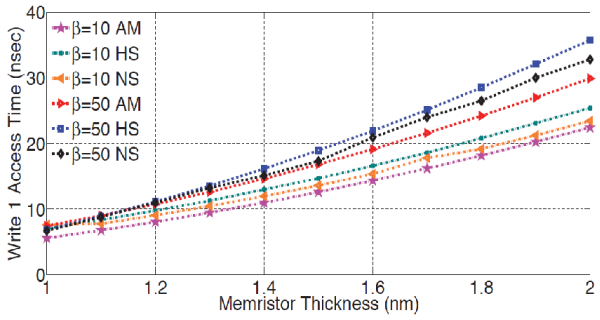

Figure 6: Analytical model vs HSPICE simulations for write 1 access time as a function of memristor thickness for different β. AM = Analytical model ignoring bitline capacitance, HS = Hspice simulations and NS = Numerical solution accounting for bitline capacitance.

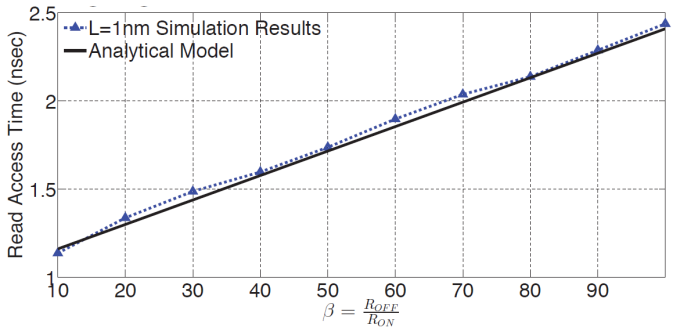

Figure 7: Read time model: Analytical model versus HSPICE simulation results for read access time of the 1T1R cell as a function of β for different memristor thicknesses L.

ing to this equation then, if the thickness of the memristor or the difference in resistance between an off state and on state increase, then the time to write will increase. If the applied voltage is increased the time to write will decrease. Using this model a read time of approx. 7 ns was found with $\beta = 10$ and $L = 1nm$ when writing a 1, but when writing a 0 with the same parameters it can be as low as 4 ns. A graph of the model is provided in Figure 6 [9].

4.1.2 Read Time Model

The equation to model the time required to read is:

$$T_R = 0.69(R_{ch} + R_{BL} + R_{Off})C_{BL}$$

R_{ch} is the resistance of the access transistor, R_{BL} is the resistance of the bitline resistor, and C_{BL} is the bitline capacitance. Figure 7 provides an illustration with R_{ON} = 100 ohms, R_{tg} = 582 ohms , and C_{BL} = 200 fF [9].

4.1.3 Write Energy Model

The model for energy consumption during a write procedure is:

$$E_w = \frac{V_{DD} I_1}{2(\zeta)}$$

I_1 is $\int_{.1}^{.9} \frac{1}{1-x^4} dx$, the integral of the inverse of the window function which models the nonlinear rate of state change of

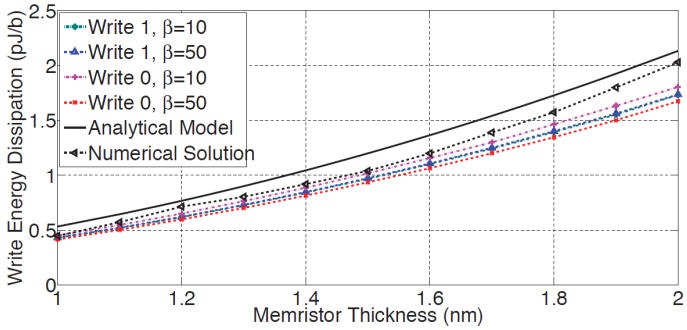

Figure 8: Write Energy Model: AM = Analytical model ignoring the bitline capacitance and NS = Numerical solution considering bitline capacitance.

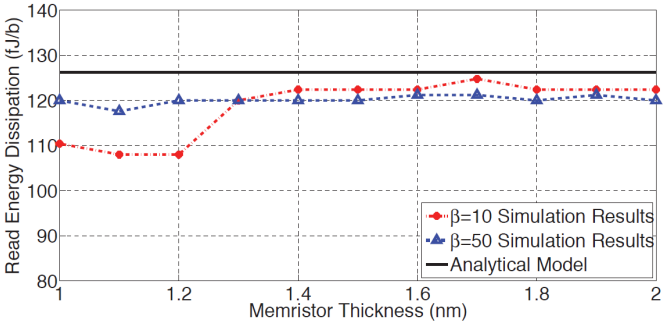

Figure 9: Comparison between the proposed model and HSPICE simulations of read energy.

a memristor and $\zeta = \frac{\mu_v R_{ON}}{L^2}$. Figure **??** graphs the model as a function of memristor thickness with ($Rtg = 582ohms$, $RON = 100ohms$, and $CBL = 200fF$) [9].

4.1.4 Read Energy Model

The model for energy consumption during a read procedure is:

$$E_R = 0.63 C_{BL} V_{DD}^2$$

Figure 9 provides a graph of the model, with R_{On} = $100ohms$, C_{BL} = $200fF$, which shows that the energy to read is independent of memristor thickness [9].

5. CONCLUSIONS

As covered in the paper, the memristor appears to be a promising new device for high density, high speed, and energy efficient computing. This paper should have also made the history of the memristor a little more understandable as it is somewhat complicated. It has been shown to be suitable for new computing architectures that do not suffer from a communications bottleneck. It also has the potential to replace many of the mediums for data storage such as NAND and DRAM as it appears to be a prime candidate as a universal memory. There is still likely much to do before memristors start replacing transistors in processors, but the potential is there.

Acknowledgments

6. REFERENCES

[1] F. Argall. Switching phenomena in titanium oxide thin films. *Solid State Electronics*, 11:535–541, May 1968.

[2] L. Chua. Resistance switching memories are memristors. *Applied Physics A*, 102(4):765–783, 2011.

[3] L. O. Chua. Memristor - the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, Semptember 1971.

[4] M. Fink. Beyond dram and flash, part 2: New memory technology for the data deluge, 2014. [Online; accessed 18-October-2015].

[5] S. Hamdioui, L. Xie, H. A. D. Nguyen, M. Taouil, K. Bertels, H. Corporaal, H. Jiao, F. Catthoor, D. Wouters, L. Eike, and J. van Lunteren. Memristor based computation-in-memory architecture for data-intensive applications. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 1718–1725, San Jose, CA, USA, 2015. EDA Consortium.

[6] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser. Memristor-based material implication (imply) logic: Design principles and methodologies. In *IEEE Transactions on VLSI Systems*, volume 22, pages 2054–2066, 2014.

[7] D. B. Strukov, X. Meng, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453:80–83, May 2008.

[8] S. Vongehr and X. Meng. The missing memristor has not been found. *Scientific Report*, 5(11657), June 2015.

[9] M. Zangeneh and A. Joshi. Performance and energy models for memristor-based 1t1r rram cell. In *Proceedings of the Great Lakes Symposium on VLSI*, GLSVLSI '12, pages 9–14, New York, NY, USA, 2012. ACM.