N x N Memristive Crossbar Array with LIF Neuron model using VHDL

Shashidhara H R[1], Mrs. Deepthi M S[2], Prof. Sai Kiran Oruganti[3]

[1 3]Lincoln University College, Malaysia; [2] The National Institute of Engineering, Mysuru;

Email ID shashidharahr@gmail.com, ms.deepthi1391@gmail.com , saisharma@lincoln.edu.

Abstract: Memristor crossbar arrays as a form of in-memory computing offers advantages in terms of density, nonvolatility and compatibility with CMOS technology. One challenge, however, is that traditional simulation methods such as HSPICE, while accurate, cannot run large-scale design verification operations in a reasonable timeframe. To address this challenge, we present a fast VHDL-based design and verification framework to support the design of memristor crossbar systems under MAGIC and FBLC logic families. This tool provides functional validation and energy, delay, timing and area overhead measurements. Performance results show that, compared to traditional simulators like HSPICE, our framework provides up to 153× speed increase in simulation execution time, with only 6% to 10% error rate for energy and delay estimates. Moreover, the framework was successfully explored through a simple 1-bit full adder block and allows for early analyses of memristor-based in-memory computing hardware implementations.

Keywords: Memristor; LIF neuron Model; NxN Crossbar array; NxN Crossbar array integrated with LIF;

Introduction

The development of artificial intelligence, machine learning, and data-driven applications has increased the demand for high-performance energy-efficient computing systems. Conventional computing systems like von Neumann model have inherent restrictions due to some physical separation of the memory and processing units. Hence, this separation is causing unnecessary data transfer, increased latency, and high-power consumption, which is known as the von Neumann bottleneck. When the data volumes increase, these limits become more apparent, which leads to the development of alternate computing models. Neuromorphic computing is inspired by the structure and functionality of the human brain. Biological brain systems do compute in a massively parallel and event-driven way by using relatively little electricity. In neuromorphic systems, the neurons process information with spikes, while the synapses store memory locally, that allows for close coupling between computation and storage. This brain-inspired technique provides good benefits in terms of parallelism, adaptability, and energy efficiency, because of their non-volatile nature, small size, and capacity to represent synaptic weights through changeable resistance states, memristive devices has got a lot of interest in neuromorphic hardware design. Memristors perform weighted summation operations when they are arranged as a crossbar array, which are the foundation of neural network computation. The crossbar structure helps in enabling simultaneous computation across multiple synapses, which makes it suitable for in-memory computing and neural network acceleration V(t)=M(q(t))I(t), However, the accurate modeling of memristive devices by using analog or SPICE-based techniques are computationally intensive and are difficult to scale for large networks. These kinds of models require complex differential equations and also long simulation times, which limits their practicality for system-level design and hardware prototyping. In order to overcome these kinds of challenges, high-level behavioral abstractions are adopted to represent memristive behavior in a simplified yet effective manner.

Key Contribution

In this project, a Leaky Integrate-and-Fire (LIF) neuron model is employed as a digital abstraction inorder to represent both the neuronal dynamics and memristive synaptic behavior. Since the LIF model captures essential characteristics such as integration, leakage, threshold-based firing, and reset behavior, making it good suited for neuromorphic applications and by utilizing this abstraction, the complexity of analog modeling is avoided and helps in retaining biological computation.

The work focuses on the design and the implementation of an N × N memristive crossbar array integrated with LIF neuron models using VHDL. This architecture enables spike-based input processing, parallel weighted summation within the crossbar, and also threshold-driven output neuron firing. Hence the entire system is developed by using synthesizable VHDL constructs and validated through simulation and FPGA implementation. This approach provides a scalable, hardware- friendly for exploring memristor-inspired

Methodology:

System for the in-memory processing and neuromorphic computing: The memristor was mainly described in-terms of a non-linear functional relationship between the quantity of electric charge which has flowed q(t), and the magnetic flux linkage, Φm(t) is given by:

$$f(\Phi_{\mathrm{m}}(t), q(t)) = 0.$$

An inductor's circuit characteristic is mainly used to generalize the magnetic flux linkage, Φm . In this case, it will not indicate a magnetic field. Below is a discussion of its physical meaning. One more way of thinking about the symbol Φm is as the integral of voltage over time. Here each memristor is defined by its memristance function, which will describe the charge-dependent rate of change of flux with charge. Since the derivative of one with respect to the other depends on the value of one or the other in the relationship between Φm and q. The practical forms are obtained by just replacing the flux with the time integral of the voltage and also the charge with the time integral of the current:

$$M(q(t)) = \frac{\mathrm{d}\Phi/\mathrm{d}t}{\mathrm{d}q/\mathrm{d}t} = \frac{V(t)}{I(t)}.$$

This suggests that memristance is the resistance that is reliant on charge. Hence, Ohm's law is obtained if M(x) is a constant function (i.e., has the same value for all x): R(t) = V(t)/I(t). The equation is not equivalent if M(x) is nontrivial since M(q(t)) and q(t) change with time. Finding the voltage as a function of time yields If M does not change with charge, this above equation shows that memristance which defines a linear relationship between current and voltage. The time-varying charge is implied by non-zero current. If the highest value of q does not significantly alter then M is relative to the beginning value M (0), On the other hand, It might demonstrate the linear dependency in circuit operation by producing a detectable voltage without net charge movement. Also, if no current is applied, then the memristor's memristance remains constant, because q(t) is constant, M(q(t)) is constant, if I(t) = 0. This is called the memory effect. Similarly, we can also define a W(ϕ(t)) as memductance, which is a portmanteau of conductance and memory:

$$I(t) = W(\phi(t)) \cdot V(t).$$

Memductance with the flux is the inverse of memristance with respect to charge, i.e.

$$\mathrm{W}(\phi(\mathrm{t})) = \frac{1}{\mathrm{M}(\mathrm{q}(\mathrm{t}))},$$

$$P(t) = I(t) \cdot V(t) = I^2(t) \cdot M(q(t)).$$

and therefore, the unit of memductance is same as the unit of conductance – Siemens. The power consumption characteristic recalls that of a resistor, I^2R:

As M(q(t)) varies little under the alternating current, then the memristor will appear as a constant resistor. If M(q(t)) increases rapidly then current and power consumption will quickly stop.

Leaky Integrate-and-Fire (LIF) Neuron Model as a Memristor: In this work, the behavior of the memristor is abstracted by using the Leaky Integrate-and-Fire (LIF) neuron model, and a well- established model in computational neuroscience which effectively captures the key dynamical properties relevant to resistive switching devices. The LIF model also emulates membrane potential integration, natural leakage, threshold-based firing, and post-firing reset behavior, all of which can be closely mapped to the physical and electrical characteristics of memristors. Specifically, when the gradual accumulation of the membrane potential corresponds to charge accumulation within the memristive device, while the leakage term models resistive relaxation and temporal decay. The Threshold firing mechanism mirrors the abrupt transition of a memristor from a high-resistance state (HRS) to a low- resistance state (LRS), and then reset operation reflects the device's ability to return to an initial state after switching. This kind of abstraction enables efficient representation of the memristor dynamics while preserving essential switching behavior. In-Memory Computing with LIF-Modeled Memristors: In-memory computing architectures, LIF-modeled memristors enable efficient simulation of computation and storage co-location in arrays of crossbars. Modelling resistive state evolution, spiking-driven computing, synaptic weight updates, and the event-driven parallel neuron activation which are made easier by LIF abstraction. This kind of method fits well with paradigms of neuromorphic computing, where the information processing is parallel and asynchronous by nature. A hybrid neuromorphic–in-memory architecture that can execute stateful logic operations, weighted summation, neural activation, and the threshold-based computation within the memory itself is achieved by incorporating LIF neurons into memristor crossbars. Compared to the traditional von Neumann systems, these architectures minimize the data travel and increase energy efficiency.

N × N Crossbar with LIF Neuron Model: The goal of neuromorphic computing is to have highly efficient, parallel, and adaptable computation by emulating the structure and operation of the biological brain networks in hardware. Large-scale neural network operations are also hindered by the memory bottleneck and high energy consumption of conventional von Neumann architectures, which are distinguished by the distinct memory and processing units, because of their high integration density, non-volatility, and inherent ability to carry out parallel analogue computations—particularly for vector-matrix multiplications, which are foundation of the neural network processing - memristive crossbar arrays have drawn a lot of interest as an alternative hardware substrate. A memristive crossbar array is made up of N horizontal word lines and N vertical bit lines, with the programmable devices (memristors) placed at each intersection. Each memristor stores the synaptic weight of a neural network and contributes to the computation of the weighted sum of neuron inputs. This crossbar naturally performs parallel weighted

summation of input signals using Ohm's law and Kirchhoff's current law by enhancing computing efficiency over sequential digital solutions.

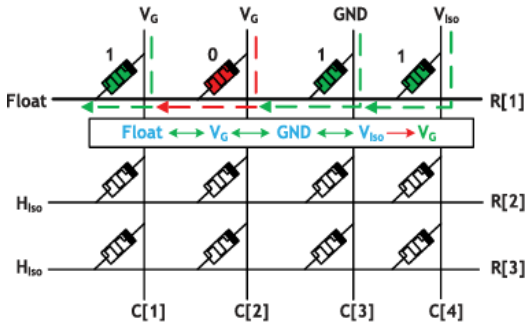


Figure 1. Crossbar array

This current work describes the detailed implementation of the proposed memristor-based crossbar array integrated with Leaky Integrate-and-Fire (LIF) neuron models, developed using VHDL. This design emulates neuromorphic computation by combining synaptic behavior using memristors and neuronal dynamics by using LIF neurons. The system is also scalable to an N×N crossbar architecture and simulated using Xilinx Vivado.

Crossbar Switch Implementation Using LIF Neuron Model: The input spikes are applied to the crossbar rows. The active memristors produce currents that flow to the output columns. The column currents are accumulated together. The LIF neurons integrate the accumulated current over time. The output spikes indicate the active output paths.

- Input spikes are applied to the crossbar rows
- Current is passed through columns
- Current from columns is summed
- Current accumulated by LIF neurons over time
- Output spikes indicate active output paths

This action mirrors how biological neural networks compute using synapses and neurons cooperating.

The design is being developed progressively from a base 2×2 crossbar to an N×N scalable architecture to validate functionality, understand system operation, and confirm scalability. At each stage, the same basic principles are utilized, and the complexity increases.

Each crossbar requires multiple spikes on the input lines, output signals from the neurons, a clock signal, a reset signal and a control signal, which adds up very quickly to pin usage as the large dimension of the crossbar increases. To ensure reliable hardware validation and ease of firmware debugging, the crossbars were implemented as 2x2 and 3x3 on the FPGA. The 2x2 crossbar was implemented as a test of new neuron firing dynamics and memristor-based connectivity. The 3x3 crossbar was able to provide increased synaptic fan-in and provide a closer representation of neuromorphic cellular behavior in a scalable way within the pin limitations of the FPGA. Model: Layer-1: Input Encoding and Synaptic Integration: In this architecture, external input spikes go into a crossbar NxN array structure. Each cross-point of the array is equivalent to a synaptic weight. When an input spike is present, electrical energy will be transferred through the crossbar based on programmed conductance values (or weights) of the synapse. All rows' current outputs are then combined and sent to a Leaky Integrate and Fire (LIF) neuron column by column for processing. Each LIF neuron will integrate the current from the previous layer over time, apply leakage

(in which a portion of charge is lost), and produce an output spike when the voltage reaches an acceptable threshold.

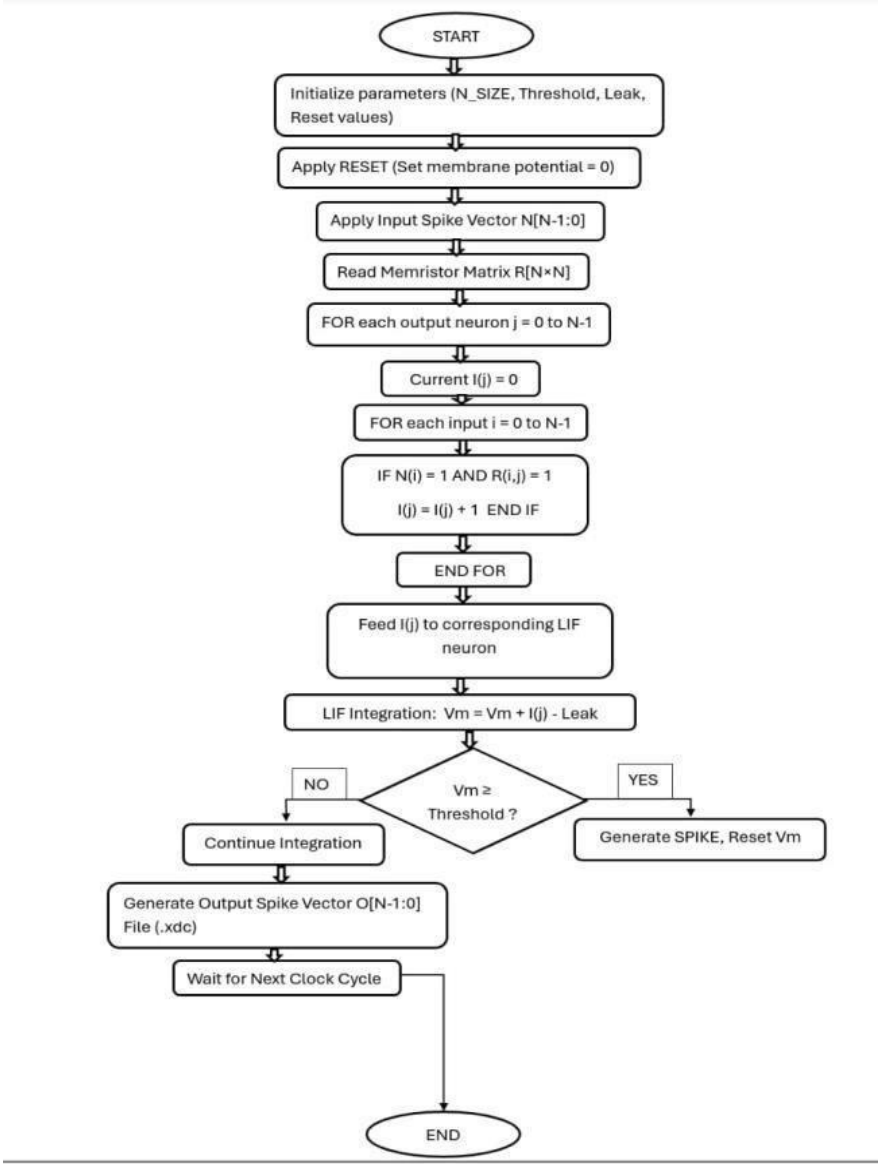


Figure 4. 2 Layer Crossbar Array Switch using LIF Neuron

Layer-2: Higher-Level Processing: The spike outputs generated by Layer-1 neurons are routed as inputs to the second crossbar layer. This layer performs another stage of synaptic weighting and integration, enabling the system to detect higher-level patterns or performance classification. The second layer neurons again follow the LIF dynamics, producing output spikes that represent the final decision or processed result. Implementation of 2 × 2 Crossbar Switch Using LIF Neuron Model: The crossbar switch is a two-cross-connect, 2 x 2 crossbar switch that has two inputs and two outputs, whereby the two input lines connect to the two output lines in the switch through programmable resistive devices. The resistive devices in the crossbar switch are modeled digitally (in a computer) and subsequently connected to LIF neuron models. The binary inputs are provided to the crossbar (input); if there is a spike, a binary (1) will be passed; conversely, if no spike is detected, the input becomes (0). The output of the crossbar is

determined by the resistive matrix R to which input is connected with which output. The output generated is equivalent to the sum of all the input(s) connected to the output by the programmable resistive devices, resulting in the same function as the synaptic weight matrix within a neuromorphic system. Role of Resistive Matrix (R = 9): The resistive matrix R is represented as a 4-bit value for a 2 × 2 crossbar. For R = 9, the binary representation is: R = 9 → 1001 This indicates: R0 = 1 → Connection between Input 0 and Output 0 is ON R3 = 1 → Connection between Input 1 and Output 1 is ON R1 = 0, R2 = 0 → Cross connections are OFF Spiking Operation for Input = 1 and 0. When the input signal is a 1, and the resistive components R0 or R3 are enabled, then the input signal of a R0 or R3 corresponds to a current being injected into the neuron. This current then accumulates over each clock cycle, causing the membrane potential (the total potential difference across the cell membrane) to increase until it reaches an established threshold level. At this point, the neuron produces an output spike (an output of 1). For an input of 0, no current will flow, and as a result, the cell membrane potential will decrease through leakage until a spike is produced.

Results: The output of the output waveforms that were produced for an active input/output cross and resistor configuration. For Example: The first "spike" occurs on Output 0 when an Input signal of 1 occurs through Rn=0. The second "spike" occurs on Output 1 when an Input signal of 1 through Rn=1. There will be no spurious spikes produced since there are no connections between the different rows and columns of the crossbar circuit. This validates that the crossbar directed the input spikes correctly and that the LIF neurons would perform their function of integrating and generating an output for input currents received.

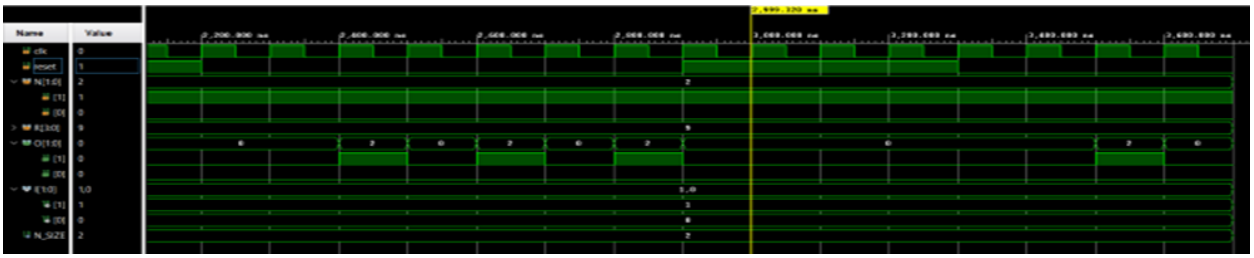


Figure 5. Output Waveform 2x2 Crossbar Switch

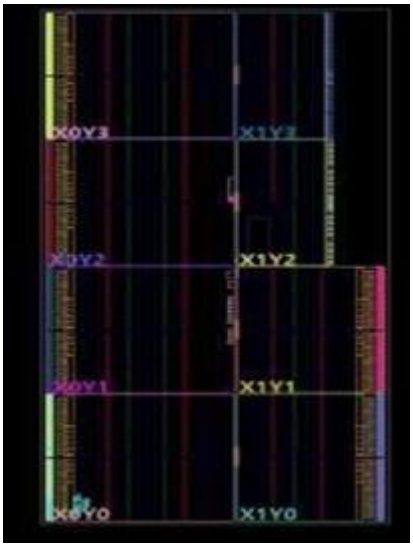


Figure 6. Schematic of 2 × 2 Crossbar Switch

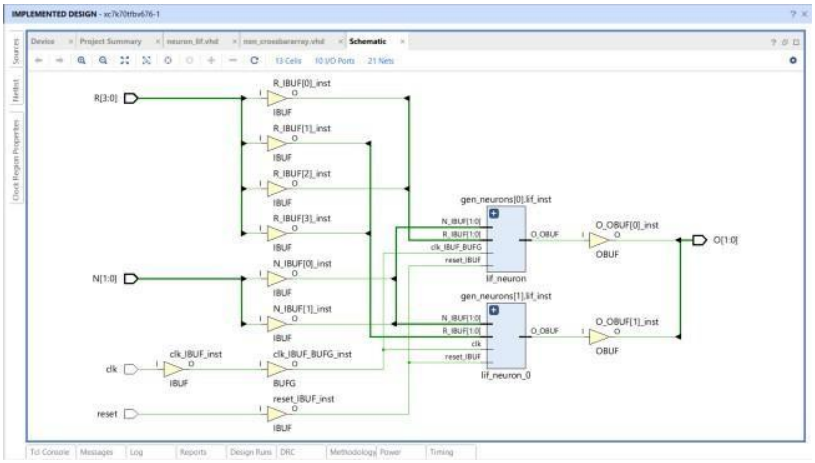


Figure 7. Implementation of 8 × 8 Crossbar Switch Using LIF Neuron Model using N x N Crossbar

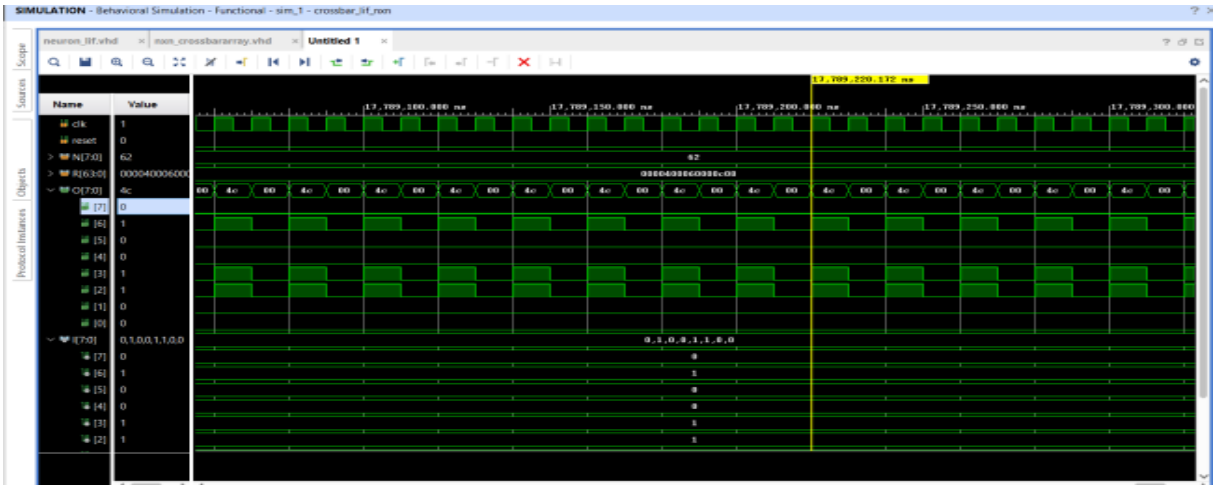


Figure 8. Output Waveform 8x8 Crossbar Switch

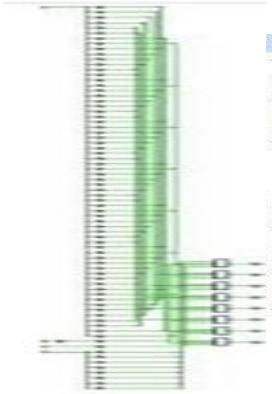


Figure 9. Schematic of 8 × 8 Crossbar Switch

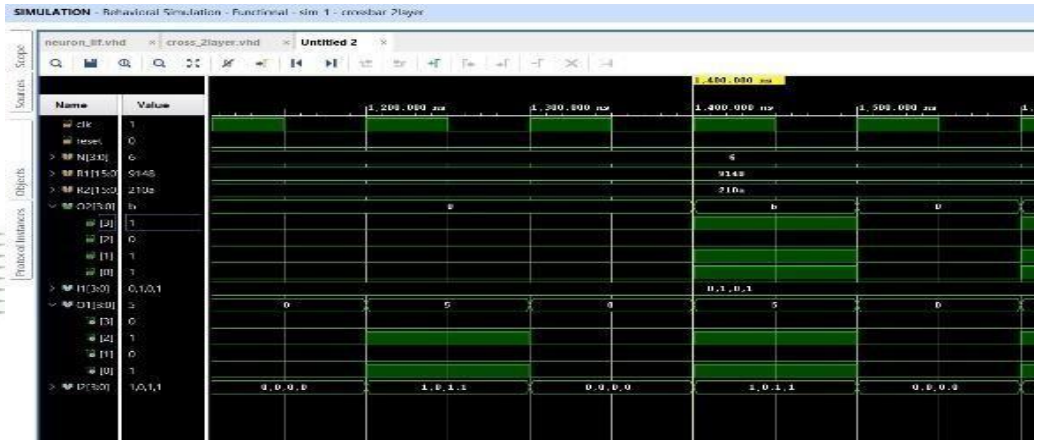


Figure 10. IV NeuroPack Simulator

Conclusion

This project presents the design and implementation of an N × N memristive crossbar array integrated with a Leaky Integrate- and-Fire (LIF) neuron model using VHDL. The proposed architecture enables in-memory and neuromorphic computation, reducing data movement and improving energy efficiency compared to conventional architecture. By using the LIF model as a digital abstraction of memristive behavior, the system achieves scalability and hardware feasibility. The design was successfully simulated in Vivado and validated on FPGA for smaller crossbar sizes. Overall, the work demonstrates an efficient and practical platform for prototyping memristor- inspired neuromorphic systems.

Reference

1. Chua, L. (1971). "Memristor-The missing circuit element". IEEE Transactions on Circuit Theory. 18 (5): 507–519. doi:10.1109/TCT.1971.1083337

2. Pershin, Y. V.; Di Ventra, M. (2019). "A simple test for ideal memristors". Journal of Physics D: Applied Physics. 52 (1): 01LT01. doi:10.1088/1361-6463/aae680. S2CID 53506924
3. Kim, J.; Pershin, Y. V.; Yin, M.; Datta, T.; Di Ventra, M. (2019). "An experimental proof that resistance-switching memories are not memristors". Advanced Electronic Materials. doi:10.1002/aelm.202000010. S2CID 202577242
4. Paul L. Penfield Jr. (1974). "1. Frequency-Power Formulas for Josephson Junctions". V. Microwave and Millimeter Wave Techniques (PDF) (Report). pp. 31–32. QPR No. 113.
5. Thompson, E.D. (1973), "Power flow for Josephson Elements", IEEE Trans. Electron Devices, 20 (8): 680–683, doi:10.1109/T-ED.1973.17728
6. Md Adnan Zaman, Rajeev Joshi, and Srinivas Katkoori. 2020. High level modeling of memristive crossbar arrays. In 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 524–529, 2020.
7. William A. Wulf and Sally A. McKee. 1995. Hitting the memory wall: Implications of the obvious. ACM SIGARCH Computer Architecture News 23, 1 (1995), 20–24.
8. Rajeev Joshi, Md Adnan Zaman, and Srinivas Katkoori. 2020. Novel bit-sliced near-memory computing based VLSI architecture for fast Sobel edge detection in IoT edge devices. In 2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS). 291–296, 2020.
9. Rotem Ben Hur and Shahar Kvatinsky. 2016. Memristive memory processing unit (MPU) controller for in-memory processing. In 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE). 1–5. 2016.
10. Shahar Kvatinsky, Eby G. Friedman, Avinoam Kolodny, and Uri C. Weiser. 2013. The desired memristor for circuit designers. IEEE Circuits and Systems Magazine 13, 2 (Second quarter 2013), 17–22.
11. Dmitri B. Strukov, G. S. Snider, Duncan. R. Stewart, R. Stanley Williams, ”The missing memristor found,” Nature, vol. 453, no. 179, pp. 80-83, May 2008.
12. Dmitri B. Strukov, Julien L. Borghetti, R. Stanley Williams, ”Coupled Ionic and Electronic Transport Model of Thin-Film Semiconductor Memristive Behavior.” Small 2009. 5, no. 9. 1058−1063.
13. http://spectrum.ieee.org/semiconductors/ processors/how- we-found-the-missing-memristor
14. Dmitri B. Strukov et al. ”Hybrid CMOS/Memristor Circuits”, Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on May 30 2010-June 2 2010, pp. 1967 - 1970, Paris
15. Hyongsuk Kim, Maheshwar Pd. Sah, Changju Yang, Seongik Cho, and Leon O. Chua, ”Memristor Emulator for Memristor Circuit Applications” IEEE Transactions on Circuits and Systems−I: Regular Papers Vol. 59, No. 10, October 2012.