Research Progress on Memristor: From Synapses to Computing Systems

Xiaoxuan Yang, *Student Member, IEEE*, Brady Taylor, *Student Member, IEEE*, Ailong Wu, Yiran Chen, *Fellow, IEEE*, and Leon O. Chua, *Life Fellow, IEEE*

***Abstract*— As the limits of transistor technology are approached, feature size in integrated circuit transistors has been reduced very near to the minimum physically-realizable channel length, and it has become increasingly difficult to meet expectations outlined by Moore's law. As one of the most promising devices to replace transistors, memristors have many excellent properties that can be leveraged to develop new types of neural and non-von Neumann computing systems, which are expected to revolutionize information-processing technology. This survey provides a comparative overview of research progress on memristors. Different memristor synaptic devices are classified according to stimulation patterns and the working mechanisms of these various synaptic devices are analyzed in detail. Crossbar-based memristors have demonstrated advantages in physically executing vector-matrix multiplication and enabling highly power-efficient and area-efficient neuromorphic system designs. The extensive uses of crossbar-based memristors cover in-memory logic, vector-matrix multiplication, and many other fundamental computing operations. Furthermore, memristor-based architectures for efficient neural network training and inference have been studied. However, memristors have non-ideal properties due to programming inaccuracies and device imperfections from fabrication, which lead to error or mismatch in computed results. To build reliable memristor-based designs, circuit-level, algorithm-level, and system-level solutions to memristor reliability issues are being studied. To this end, state-of-the-art realizations of memristor crossbars, crossbar-based designs, and peripheral circuitry are presented, which show both promising full-system inference accuracy and excellent power efficiency in multiple tasks. Memristor in-situ learning benefits from high energy efficiency and biologically-imitative characteristics, which are conducive to further realizing hardware acceleration of cognitive learning. At present, the learning and training processes of brain-like networks are complex, presenting great challenges for network design and implementation.**

Manuscript received December 17, 2021; revised February 14, 2022; accepted March 9, 2022. Date of publication March 21, 2022; date of current version April 28, 2022. The work of Ailong Wu was supported in part by the Natural Science Foundation of China under Grant 61976084 and in part by the Natural Science Foundation of Hubei Province of China under Grant 2021CFA080. This article was recommended by Associate Editor M.-F. Chang. *(Xiaoxuan Yang and Brady Taylor contributed equally to this work.) (Corresponding author: Yiran Chen.)*

Xiaoxuan Yang, Brady Taylor, and Yiran Chen are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: xy92@duke.edu; brady.g.taylor@duke.edu; yiran.chen@duke.edu).

Ailong Wu is with the College of Mathematics and Statistics, Hubei Normal University, Huangshi 435002, China (e-mail: hbnuwu@yeah.net).

Leon O. Chua is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: chua@berkeley.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2022.3159153.

Digital Object Identifier 10.1109/TCSI.2022.3159153

***Index Terms*— Memristor, memristive devices, processing-in-memory, reliability, in-situ learning, neuromorphic computing.**

I. Introduction

MEMRISTORS were originally proposed by Prof. Leon Ong Chua in 1971 as a fundamental circuit device representing the relationship between magnetic flux and charge [1]. In 2008, researchers from HP Labs created a nanoscale memristor device for the first time and provided corresponding mathematical models and circuit simulations matching Prof. Chua's descriptions [2]. Due to its high density, low access energy, and nanoscale dimensions, the memristor has been regarded as one of the most promising emerging non-volatile memory technologies. Utilizing memristors to store data is area-efficient due to their ability to store multiple bits per cell. Programming and reading data from memristors is achieved through applying voltage across the memristor above and below a threshold, respectively. Furthermore, memristor cells can be integrated into memristor arrays, crossbars, and cubes (through 3D integration [3]) for improved density.

In addition to data storage functions, memristor crossbars can realize vector-matrix multiplication (VMM). With analog voltages applied to each row, the memristor crossbar can accumulate currents along each column [5]. According to Kirchhoff's law, the output current for each column is the sum of products of voltage applied at each row multiplied by memristor conductance at each row. Therefore, the crossbar can compute using the memory itself, which significantly mitigates the communication burden of data for each operation. Fig. 1 gives an example of memristor-based vector-matrix multiplication. Memristor crossbar-based designs have been popular in many recent neural network architectures [6]–[8] due to their ability to implement VMM operations with high energy efficiency.

However, the non-idealities of memristor devices are of great concern for developing reliable memristor-based architectures. Errors in conductance stored by memristor cells, for example, cause errors in vector-matrix multiplication results. Memristor reliability issues include device faults, device endurance, IR-drop, variation, and noise. Therefore, effective circuit-level, algorithm-level, and system-level solutions are necessary to mitigate reliability issues and enable the reliable design of memristor-based neuromorphic computing systems.

Based on the discussion of memristor crossbar-based designs, this survey also presents state-of-the-art realizations

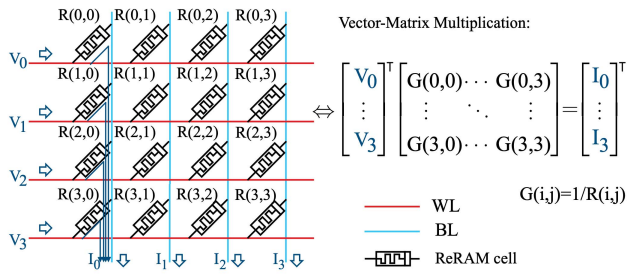


Fig. 1. Memristor-based vector-matrix multiplication [4].

of memristor crossbars and peripheral circuit designs, demonstrating both promising full-system inference accuracy and high power efficiency for many workloads.

Memristor-based in-situ learning networks feature outstanding energy efficiency and biologically-imitative qualities, making them ideal for furthering the hardware acceleration of cognitive learning. However, the training processes and learning of brain-like networks are highly sophisticated. Therefore, there are significant obstacles to the design and implementation of memristor-based in-situ learning networks.

In this survey, perspectives on the state-of-the-art research progress surrounding the memristor are provided, from synapses to computing systems. The organization of this paper is as follows: In Section II, the benefits of memristor synapses and specific synaptic structures are introduced. In Section III, the details of memristor crossbar-based designs are analyzed, particularly for in-memory logic, vector-matrix multiplication, and neural network architectures. The causes and potential solutions for memristor reliability issues are explained in Section IV. In Section V, the state-of-the-art of memristor crossbar realizations and performance are presented. In Section VI, the hardware and software considerations of memristor-based in-situ learning networks are discussed. Finally, Sections VII and VIII discuss challenges and possible future work in the study of memristors.

II. Memristor Synapses and Neurons

The synapse is the unit where biological neurons connect with each other and transmit information. The human brain contains more than 10 million neurons, each of which is connected to more than 2,000 synapses. Hence, the design and implementation of synaptic circuits are very important in the construction of artificial neural network hardware. However, traditional synaptic simulations use transistor circuits, which have two main disadvantages: state volatility and the difficulty of repetitive programming, which limit applications of transistor synapses in large-scale artificial neural networks.

The emergence of physical memristor devices opens up new possibilities for synaptic hardware simulation. Many neural circuits consist of three basic processing modules: Multiplication, accumulation, and activation. Synthesizing the weighted signals and activating the outputs of multiple synapses are key to neural circuit design. Memristors are especially suitable for realizing synapses in artificial neural circuits for the following reasons [9]: Firstly, the conductance of a memristor can represent synaptic weight and can be changed by different applied voltages. Secondly, memristors are simple two-terminal devices, which can realize high-density, integrated crossbar-array structures. In addition, their nanoscale size, low power-consumption, and non-volatile memory properties make the structure of the memristor synapse smaller and more efficient than conventional transistor synapses.

Expanding on the fundamentals of memristor synapses, researchers have proposed different types of synapse structures, including five-memristor synapses [10], one-transistor and one-memristor (1T1M) synapses [11], and two-transistor and one-memristor (2T1M) synapses [12]. In order to achieve efficient synaptic operation, many different memristor synapses have differing topologies, such as memristor bridge circuits and memristor/double-memristor synaptic arrays. The former has a complex structure, which is not convenient for large-scale integration. In the latter, because the structure of memristors in the array is intersect-linked and fixed, the corresponding row or column of the array must be selected in advance when programming a memristor weight. This makes the relevant circuitry unable to achieve full parallel weight programming. Therefore, further work to construct new circuits to achieve large-scale parallel weight adjustment is required. Chen *et al.* [10] proposed a synaptic circuit composed of five memristors. This circuit can execute positive and negative synaptic weights, in which only one memristor is used for weight calculation, and the other four memristors are used for symbol-setting. In order to realize bidirectional control of a single memristor, Dubey *et al.* [12] proposed a 2T1M synaptic structure, which uses two CMOS switches to control the input of positive and negative signals. Furthermore, some memristor synaptic structures have been proposed to simulate spike-timing-dependent plasticity (STDP) in spiking neural circuits, which further verifies the advantage of memristors in simulating synaptic learning for brain-inspired computing [13].

Another widely used memristor synaptic structure is the memristor crossbar array, in which a synaptic weight is expressed by the conductance of memristors connecting each crossing of a word-line and orthogonal bit-line. Through the electrical characteristics of the memristor, the array can easily execute large-scale matrix multiplication, which can be used to speed up the operation of neural networks. In order to independently control the write/read operation of each memristor in the crossbar array, many 1T1M array structures have been proposed. Given that a single memristor can represent positive synaptic weights only, Ai *et al.* [14] proposed double-memristor synaptic crossbar arrays to represent positive and negative synaptic weights, and kernel weights are defined as the element-wise difference in conductance between the two memristor arrays. CMOS-memristor hybrid technology can achieve fully-parallel programming operations that classical crossbar arrays cannot. These crossbar circuits combined with the inherent computing characteristics of memristor arrays can achieve large-scale feedforward parallel computing that significantly improves computing speed compared to serial computing techniques.

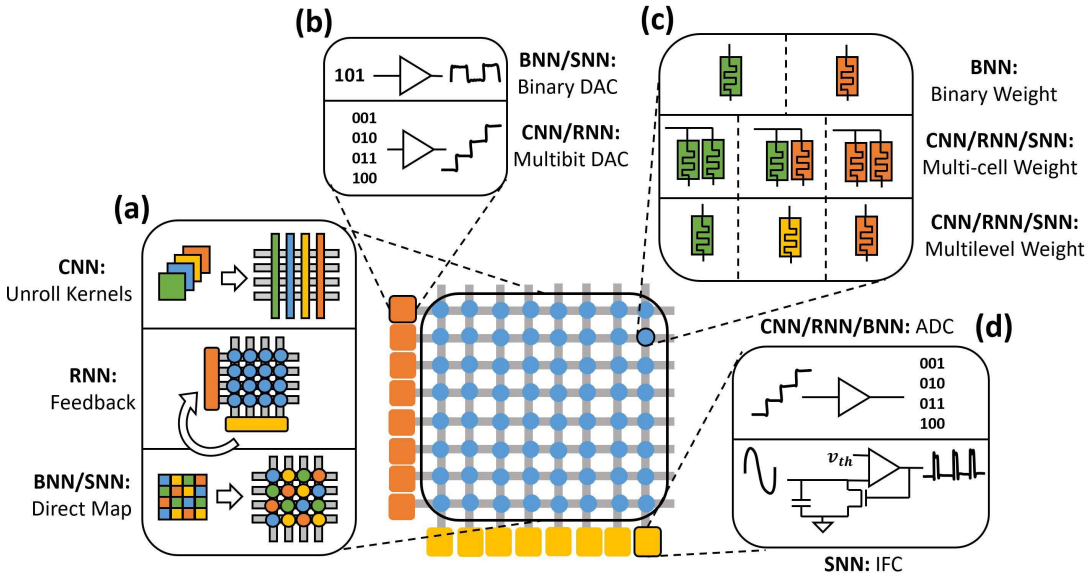


Fig. 2. Breakdown of major components in memristor crossbar-based applications: (a) Data is mapped in many ways to facilitate different applications. In the top section, CNN weight kernels are unrolled and mapped to crossbar columns as conveyed by the colored bars. In the middle section, output data is fed back to input drivers in RNN architectures. In the bottom section, weights are directly mapped to each cell of the crossbar as conveyed by the colored boxes. (b) Input drivers can include simple buffers for generating binary pulses for BNNs/SNNs, and DACs for converting digital inputs to analog voltages. (c) Memristors can represent a binary weight when in either low- or high-resistance states (LRS/HRS). They can also be tuned to specific resistances in multilevel weights, or multiple memristors across several columns can represent a single, high-precision multicell weight. (d) In the output stage, ADCs can convert current or voltage outputs into digital signals, or IFCs can be used in SNNs to convert current to spiking voltage outputs.

III. Crossbar-Based Memristor Designs

Memristors can be naturally used for vector-matrix multiplication (VMMs). A memristor-based crossbar array has word-lines (WLs) and bit-lines (BLs) that are orthogonal to each other. An input vector is encoded as analog voltages and applied to the crossbar, while the matrix is represented as conductances of the memristors and stored at each cross-point. According to Kirchhoff's current law, the VMM result is equal to the current through each crossbar column [15]. With the help of peripheral circuitry, memristor crossbar arrays support processing-in-memory (PIM) operations, including in-memory logic and memristor-based VMM. In this section, different types of memristors are examined. Then, the use of crossbars in various computing tasks is discussed. Fig. 2 provides a summary of memristor crossbar applications, including the cell design and peripheral circuits used for different neural networks.

A. Types of Memristor

Basic and common cells for memristor crossbars are categorized into one-memristor (1M) [16]–[18], 1T1M [11], [19], [20], one-selector and one-memristor (1S1M) [21]–[23], and one-diode and one-memristor (1D1M) [24], [25]. Besides these, there are one-transistor and multiple-memristor (1TnM) [26] as well as 2T1M [27] designs.

Utilizing 1M for each cell to construct the memristor crossbar demonstrates high area density and high power efficiency [17]. However, sneak path (i.e., the undesired current flowing through unselected cells) can disturb the accurate programming and reading of cells in the crossbar [28]. To suppress the sneak path problem, sensing methods with precise voltage control [29] and reading schemes with reference to the sneak current [30] have been proposed. Outside of these algorithmic approaches, the memristor can be integrated with other components to avoid sneak path problems. For instance, 1T1M, 1S1M, and 1D1M can be viewed as potential candidates for reliable memristor computing. Self-selective and self-rectifying memristors also show great potential for solving the sneak path current issue with a simple structure [31].

1T1M cells integrate a transistor with a memristor and are thus highly compatible with the CMOS process. Furthermore, 1T1M arrays can support the parallel programming of memristor cells in a row or column. More importantly, the gate in the transistor can precisely control linearity and symmetry in weight programming. However, the larger cell area compared to 1M designs constrains area efficiency. Moreover, when considering high-dimension integration, the 1T1M design also has sneak and leakage current issues [32].

1S1M designs, with the same area consumption as 1M designs, have high area efficiency and are potentially scalable to 3D. Selectors with high non-linearity are preferable for achieving high power efficiency and programming accuracy.

1D1M designs integrate a diode with each memristor. However, this design cannot program bipolar memristors since the reverse current of a diode is very small.

B. Crossbars for In-Memory Logic

Binary memristor crossbars can be used to implement in-memory logic [33], [34]. Memristor-aided logic (MAGIC) [33] is the first work of its kind to support in-memory logic with memristor devices, and it analyzes the range of the voltage needed to guarantee successful switching behavior for each logic operation. Here, an example of the operation of

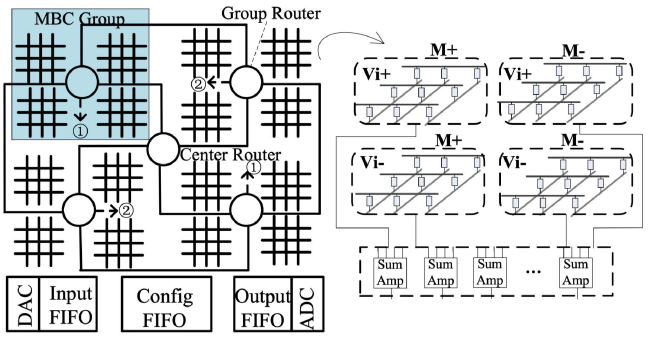


Fig. 3. Memristor-based reconfigurable neuromorphic computing accelerator architecture [38].

implementing NOR logic with memristors is given. Firstly, the input cells, as well as the output memristor cell, are all placed in the same row. Input cells starting with unknown status will be programmed to store the input values as conductance states. In the NOR implementation, the output cell is initialized to high resistance first. In the first cycle, the p terminals of the input cells are set to an execution voltage, while the p terminal of the output cell is grounded. As a result, the output cell will switch to a low resistance if there exists at least one high resistance cell among input cells. This NOR operation only takes a single cycle. Following the FELIX design [34], memristor-based logic for INV, OR, and XOR operations can be executed within one, two, and two cycles, respectively.

Single memristor-based logic gates (such as NOR) can be further utilized to fulfill more complex functionalities, such as addition [35] or multiplication [36]. Similarly, digital processing-in-memory designs can be built with memristors. For instance, Imani *et al.* [37] proposed a digital memristor-based computing system named FloatPIM, which enables floating-point representation and calculations with single-bit memristors. Moreover, Yang *et al.* [4] exploited memristor-based in-memory logic techniques to implement a memristor-based hybrid softmax design. While memristor-based in-memory logic benefits from substantial parallelism by executing the same operation on many rows simultaneously, the minimization of data transfer, and the very high density of memristor arrays, its practicality suffers from the low endurance of memristor technology to reprogramming resistive state, as well as the duration required by multiple write operations for a single logic operation. The endurance issue will be further addressed in section IV.

C. Crossbars for Vector-Matrix Multiplication

Hu *et al.* [5] first proposed the use of memristor crossbars for vector-matrix multiplication (VMM). To support VMM calculations, the network's weights and activations should be mapped to each memristor's conductance and crossbar input voltages according to the resolution of the memristor devices and digital-to-analog converters (DACs), respectively [38]. A design of memristor-based reconfigurable neuromorphic computing accelerator (RENO) architecture is shown in Fig. 3. When memristor cell resolution is lower than the weight quantization bit number, multiple cells can be used to represent a single weight parameter. Since the memristor conductance can only represent positive values, the difference of the results from two arrays is utilized to represent both positive and negative values [39]. A kernel in a convolutional layer needs to first be unrolled and mapped to a crossbar, while a kernel in a fully connected layer is mapped directly to a crossbar [6]. The memristor computation process accumulates the calculated results from crossbars and uses analog-to-digital converters (ADC) to generate outputs. Note that VMM operations in memristor crossbars do not require repeated reprogramming of memristors as for in-memory logic and therefore do not suffer from endurance issues, but repeatedly "reading" these memristors during VMM operations can slowly alter device state. This issue is further addressed in section IV.

D. Crossbars for Neural Network Architectures

Memristor-based crossbars have become indispensable for implementing neural network architectures in the analog domain. As crossbars provide an efficient and nigh-instantaneous means of physically emulating VMM operations, different neural network algorithms such as convolutional, spiking, and recurrent neural networks can be executed by architectures that process and transport data in the form of voltages or currents between memristor crossbars. In this section, architectures for convolutional, spiking, and recurrent neural networks are briefly explored, as well as the methodologies and physics that enable them.

1) Convolutional Neural Networks: Convolutional neural networks (CNNs) are by far the most readily explored derivative of neural network algorithms for memristor-based architectures due to the ability of CNNs to efficiently extract the spatial information of data. PRIME is one of the earliest crossbar-based architectures for neural networks capable of implementing either multilayer perceptrons (MLPs) or CNNs [7]. CNN kernels are trained offline and mapped to crossbars, with both positive and negative weights represented by the column-wise difference in outputs of two separate crossbars. PRIME uses multiplexers that attach the columns of the crossbar to computational logic or write/read logic, depending on whether the array is enabled for memory or computational mode. The WL drivers can drive different analog voltages in either mode. In computational mode, the column currents for the positive and negative arrays are passed through analog subtractor and sigmoid circuits. The digital output is generated by converting the column current to a voltage and comparing the voltage to the voltage sources. The sign of the output is stored as the MSB of the output, which can be used as the select signal of a ReLU activation function if desired. PRIME brings further attention to the issue of activation and parameter precision, pointing out that both are factors in the precision of output data. While activation precision depends on the circuitry driving the crossbar array, parameter precision can be achieved either through using multiple single-bit memristor cells to represent a single parameter or through multilevel memristors [40], [41].

ISAAC expands on the concept of crossbar-based CNN architectures [8] by incorporating pipelining. The architecture

uses a network of tiles in which each tile has several crossbar units along with ADCs, shift-and-add logic, and registers. As kernels in CNNs only work on portions of the feature map at a time, there is no need to wait for all outputs of a layer to be produced before beginning to process the next layer. Furthermore, resources within each tile can be pipelined so that the next slice of a feature map can be sent to a crossbar without waiting for the previous slice to finish the conversion to a digital value. Shift-and-add logic is also necessary if kernels map to more than a single crossbar.

PipeLayer [6], beyond including training of kernels into the architecture, changes up the pipeline of ISAAC and deeply explores the mapping of kernels and activations to crossbars. Kernels are reshaped into 1-D vectors and are mapped to single columns of multiple arrays (represented once again by separate positive and negative arrays, as well as being split amongst multiple arrays if the kernel is larger than the height of a single array). The feature map is split into 1-D vectors that are applied to the rows of the arrays to generate dot-products with each kernel. While fast, this method comes at the expense of replicating data in the feature map. Each vector of the feature map must be applied to the kernels one at a time, but duplicating the kernels with identically-mapped crossbars can accelerate the processing of the feature map at the expense of utilizing more arrays. Other works, such as MobiLattice [42], show the benefits of using mixed-signal processing to minimize the under-utilization of crossbar arrays, while works like 3D CNN [43] utilize 3D crossbars for an even denser realization of CNN architectures.

2) Spiking Neural Networks: Spiking neural networks (SNNs) refer to the encoding of input and output activations of a neural network as voltage spikes. Popular digital SNNs exist commercially, including IBM's TrueNorth and Intel's Loihi [44] [45]. Work in this area of crossbar-based neural networks largely involves the design of peripheral circuitry. PipeLayer is a partial implementation of an SNN [6], using an integrate-and-fire circuit (IFC) to convert an analog current into a spiking voltage signal, and then into a digital output. The IFC is a staple of crossbar-based SNN implementations [46]. In an IFC, the current from the column of a crossbar is integrated by a capacitor into a proportional voltage. When this voltage reaches the predetermined threshold voltage, a comparator's output rises, activating a transistor in parallel with the capacitor to discharge it. The comparator's output falls back down, creating a spiking signal that can be buffered as input to subsequent layers, or, in the case of PipeLayer [6], a digital counter can count the spikes produced during the compute time to estimate the IFC's spiking rate.

Several aspects of this circuit can be optimized for improved performance and efficiency. To ensure that the current from the column doesn't change as the capacitor of the IFC charges, a current amplifier can be used to hold the voltage at the column constant, thereby charging the capacitor linearly [39]. This optimization is compatible with designs where two arrays are used to represent positive and negative parameters, using the amplifier to both subtract the currents and hold the column voltage constant. The addition of a buffer between the output of the IFC's comparator and the transistor in parallel with the capacitor can introduce a nonlinear activation function akin to a hyperbolic tangent function to the output spiking signal [47]. The delay of the buffer causes the spiking rate of the IFC to saturate to a value based on the current amplifier's reference voltage and the IFC's threshold voltage, such that the output spiking rate of the IFC increases linearly with column current up until the saturation rate. As for the input circuitry of an SNN, the input activation of an SNN can also take several different forms, including a simple analog voltage as seen in PipeLayer [6], another spiking waveform (as in TrueNorth [44] and Loihi [45]), or a pulse-width modulated (PWM) signal [48]. In the latter case, the signal is represented by an input signal where the duty cycle of a pulse corresponds to the digital input value. This approach benefits from higher precision capabilities and use of efficient digital circuits instead of expensive DACs.

3) Recurrent Neural Networks: Recurrent neural networks (RNNs) differ from other models in that the output depends upon both current inputs and previous outputs. For one implementation of an RNN [49], two arrays are used. The first array is split into two sections of WLs, with the current from both sections being aggregated by the hidden layer neurons. The first section of WLs is driven by the input neurons, while the second section of WLs is driven by feedback connections from the hidden layer neurons. Finally, the second array is also driven by the hidden layer neurons, connected through the crossbar to the final output layer neurons. In the neuron circuit, an integrator converts the current to voltage and is modulated by an activation function implemented via a voltage follower circuit. However, this architecture implements a vanilla RNN only, an algorithm that can suffer from the vanishing gradient problem during training. In a newer architecture [50], long short-term memory (LSTM) and gated recurrent units (GRUs), which avoid this problem, are implemented. These algorithms require element-wise multiplication and different types of activation functions (hyperbolic tangent and sigmoid), both of which are executed digitally. This architecture pipelines the crossbar-based VMM, the activation functions, and the element-wise multiplication in three separate stages and can therefore process three separate sequences simultaneously. Note that elements of the same sequence cannot be in the pipeline simultaneously, as processing one element requires that the previous element has already been processed. In another architecture [51], the VMMs are executed by the crossbar as usual, and software is used for activation functions and element-wise multiplication of the data. Other LSTM designs [52] [53] [54] opt for a completely analog design in which an analog activation circuit is used and the element-wise multiplication is performed by a voltage multiplier circuit.

4) Binary Neural Networks and Other ML Applications: Through the various neural networks mentioned above, memristor-based processing-in-memory designs show their potential in accelerating networks with high energy efficiency. However, when implementing these complex networks, memristors with multilevel states are preferable, while multiple memristor cells may be required to represent one full-precision weight value in the network. This multilevel cell is more

vulnerable to noise, and reliability issues will be discussed in the next section. Furthermore, the integration of multiple cells will increase the energy and area overhead. Therefore, memristor-based designs can further boost their performance when executing hardware-friendly networks, such as binary neural networks (BNNs) [55]–[58]. Tang *et al.* [55] proposed using a memristor crossbar as a processing engine to implement a BNN and discussed the matrix splitting problem when supporting large weight kernels. Sun *et al.* [56] put forward a parallel XNOR-RRAM architecture with a bit-cell design and enabled parallel read-out mode to improve performance. Song *et al.* [57] proposed a complementary resistive cell design for XNOR activation, and thus enabled in-situ BNN computations in memristors. Other forms of crossbar-based machine learning and neural network algorithms have also been developed, including reinforcement learning [59], generative adversarial networks (GANs) [60] [61], natural language processing (NLP) [4], and algorithms that utilize the sparsity of feature maps (such as in deconvolution) for efficient inference [61] [62]. Other general-purpose memristor-based processing-in-memory designs provide energy-efficient solutions in genome sequence alignment [63]–[66], graph processing [67]–[69], and scientific computing [70]–[72].

A simulation framework for memristor-based crossbar systems is essential to evaluate the performance, energy, and area consumption under neural network tasks. Representative frameworks include NVSim [73], MNSim [74], NeuroSim [75], RxNN [76], Pytorx [77], and the analog in-memory computing toolkit [78]. These frameworks enable fast, accurate, and flexible designs and validate the performance considering the overall architecture.

IV. Reliability Issues in Memristors

Memristor-based designs have demonstrated their potential in enabling fast and efficient neuromorphic computing systems. However, one key challenge in building trustworthy memristor-based designs lies in the non-idealities of memristor devices. For instance, faults in memristor cells introduce errors to vector-matrix multiplication results. Worse still is that the effects of device faults can accumulate through the computing system and result in serious accuracy degradation for deep neural networks. In this section, reliability issues pertaining to memristor crossbars are discussed, including device faults, device endurance, IR-drop, variation, and noise. Based on the analysis of each cause, effective circuit-level, algorithm-level, and system-level solutions have been provided to mitigate these issues and enable the reliable design of memristor-based neuromorphic computing systems. Fig. 4 provides an overview of memristor reliability issues.

A. Device Faults

Memristor device faults caused by fabrication defects and programming processes can be classified into soft faults and hard faults [79]. Soft faults refer to the errors that happen when the memristor conductance differs from the target conductance and can be fixed by re-writing or tuning the memristor conductance. Hard faults refer to the errors in which the memristor state is stuck at a low conductance (i.e., stuck-at-0, SA0) or high conductance (i.e., stuck-at-1, SA1) and cannot be further changed. Therefore, hard faults in memristors are more detrimental to reliable computing systems since they cannot be easily fixed by writing operations [80]. Methods for tolerating hard faults in memristor devices consist of the remapping strategy [79], fault-aware retraining [79], [81], redundancy columns utilization [81], [82], error-correction codes [83], [84], profiling with error compensation [77], and circuit techniques [85]. For instance, Xia *et al.* [79] observed that a neural network has sparsity in its weights, which can be matched to the memristor cells with SA0 faults. Therefore, they suggest reordering neurons to circumvent the SA0 fault. He *et al.* [77] proposed an error correction algorithm to compensate SA0 and SA1 faults, which requires one-time profiling to achieve high inferencing accuracy.

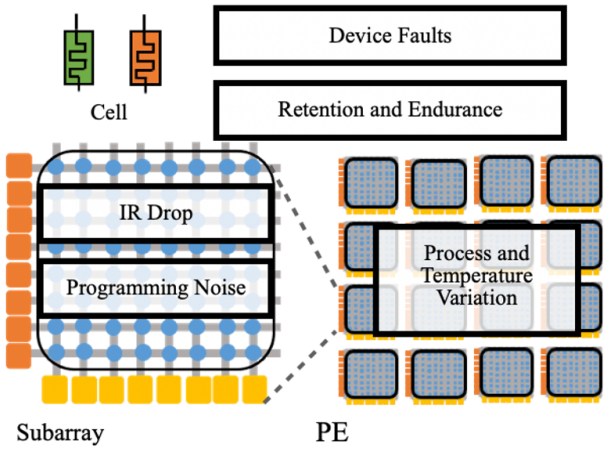


Fig. 4. Overview of memristor reliability issues. Cell level: device faults, retention and endurance issues; Subarray level: IR drop and programming noise issues; processing element (PE) level: process and temperature variation.

B. Retention and Endurance

Memristor devices have the property of high retention time for up to 10 years in some cases [86]. However, they can also experience a memristance degradation issue caused by the oxygen vacancy profile changing due to oxygen diffusion [87]. Chen *et al.* [88] proposed tuning the cycling conditions to minimize filament degradation during usage. Fukuyama *et al.* [89] investigated the optimal combination of writing techniques, including "verify", "finalize verify", and "relaxation effect". Furthermore, the retention problem is closely related to the endurance problem, as there exists a trade-off between these two issues.

Currently, the endurance of a memristor device ranges from 10^6 to 10^{12} [3], [90]–[93]. The endurance problem (also referred to as wear-out or the aging problem) sets an upper bound for reprogramming times and limits the usage of memristor devices in online training. To reconcile the endurance problem with the programming approach, Wen *et al.* [94] proposed using an optimized programming order with reset operations followed by set operations and shortening the reset operations for less significant bits. Zhang *et al.* [95] also put forward a software approach to skew trained weights toward smaller values and a hardware mapping

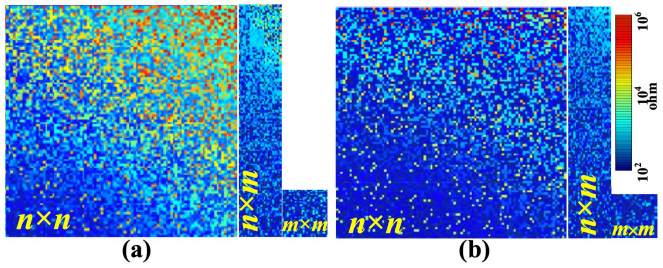


Fig. 5. Trained memristor resistance discrepancy: (a) without IR-drop compensation. (b) With IR-drop compensation [97].

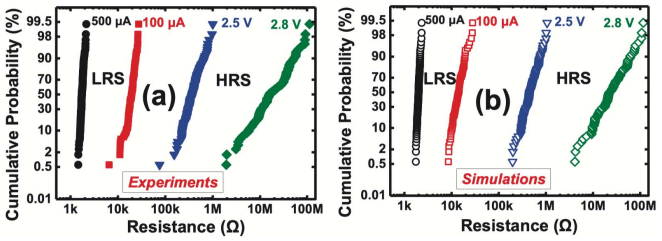


Fig. 6. (a) Measured and (b) simulated complete multilevel cell characteristics with resistance distributions due to cycle-to-cycle variations [98].

approach that considers the aging status of the memristors. Shirinzadeh *et al.* [96] utilized an endurance management framework to balance writing traffic and enhance the potential usage of a memristor-based computing architecture.

C. IR-drop

IR-drop is introduced by wire resistance. Liu *et al.* [97] analyzed the IR-drop phenomenon in various-sized memristor crossbars and put forward reduction and compensation techniques to improve the training and recall performance. As the crossbar size increases, the impact of IR-drop becomes more severe, resulting in performance variations or functional failure in the memristors. Therefore, the 1-D reduction method is put forward to utilize small crossbars for area efficiency and reliability. A large weight matrix (of size $n \times m$) can be decomposed into two smaller matrices (of size $n \times r$ and $r \times m$, where $r \ll m$). The compensation method can also update weights in the recall stage to compensate for the effect of IR-drop. Fig. 5 shows the effectiveness of IR-drop compensation, with the memristor resistance discrepancy minimized. He *et al.* [77] proposed a training method that incorporates the current shift in memristor crossbars due to IR-drop as stochastic noise in the training process.

D. Programming Noise

The programming noise in a memristor can be estimated by a Gaussian distribution or log-normal distribution [98], as shown in Fig. 6. The variation of the programming noise is related to the programmed value (i.e., memristor conductance). The programmed states can be uniformly distributed or nonuniformly distributed [99] among the available conductance range. Since the memristor noise margin is inversely proportional to the number of conductance levels, using a large-resolution memristor cell (such as an 8-bit cell) can be a challenge for the inferencing accuracy as suggested in [100]. Furthermore, multiple memristor cells can be integrated to represent a single weight if the memristor cell's resolution is smaller than the weight's resolution. In this case, programming noise will be multiplied by a scalar during integration, and the programming noise in the most significant bits is of greater concern. Therefore, a dynamic fixed-point representation [101] is utilized to remove the most significant bits which represent zero.

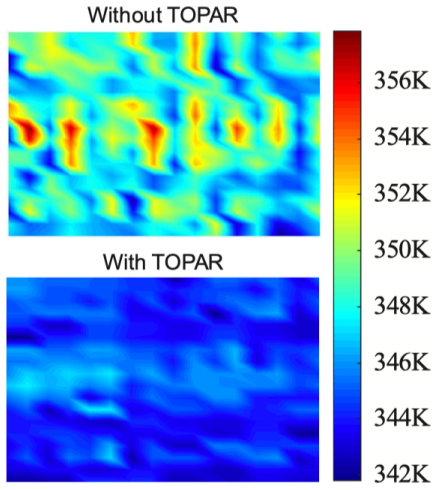


Fig. 7. Thermal distributions of memristor arrays when running the conv2_blk of ResNet50 both with and without the thermal-aware optimization framework TOPAR [107].

E. Process and Temperature Variation

In a memristor-based computing system, process variations are inevitable due to the nanoscale size of the technology [80]. To combat process variations, the sensing strategy can select and activate a limited number of rows in one cycle and provide a large sensing margin [102]. Besides, on-chip ADC reference generation is helpful for mitigating the offsets generated by the process variations [103]. Other than the circuit-level designs, training and mapping technology has been put forward to solve the process variation problem. Hu *et al.* [5] proposed a hardware training method to minimize such problems caused by process variations, where the parameter tuning strategy is based on the difference between the target value and the output value with noise. Yan *et al.* [104] observed that the memristance drift pattern is related to the conductance value and the sensing voltage direction. To mitigate the weight drift, a closed-loop circuit design is proposed that minimizes the difference between the previous recall result and the current result. Chen *et al.* [105] came up with the bipartite-matching mapping method to decrease the impact of large variations in memristors. Moreover, they design a training algorithm to leverage the inherent self-healing capability of a neural network to further prevent large weights from being mapped to a large variation memristor.

Temperature variations, also referred to as thermal noise, is generated due to the thermal agitation of the charge carriers inside the conductor [106]. Shin *et al.* [107] introduced a thermal-aware optimization framework named TOPAR, as shown in Fig. 7, to minimize the temperature variation among memristor arrays. Yang *et al.* [100] proposed a hardware-aware training method to tackle the thermal noise in a memristor-based crossbar. Even at relatively high

temperatures (such as 400 K), the inferencing accuracy of memristor-based designs can be improved.

V. Implemented Designs

In previous sections, examples of concepts and designs that illustrate the basis for memristor crossbar-based designs have been shown. In this section, state-of-the-art realizations of crossbars and accompanying designs are explored. The demonstration results show that memristor-based designs have great advantages in computing efficiency.

A. Crossbar Implementations

Memristor crossbars have made several leaps in progress in terms of speed, size, power consumption, and precision. In a recent review by Xia and Yang [28], a few notable accomplishments in memristor technology are cited that have large implications on implementing a memristor crossbar-based system. Memristor crossbars can be implemented with memristors that are 2 nm on a side and separated by about 12 nm between cells in the crossbar array [108]. Another crossbar with larger memristors (about 50 nm) is capable of switching from 0.73 to 97.83 kΩ or back in 85 ps, using only about 15 μA [109]. In terms of precision, one work reports over 500 possible conductance states [110], allowing for about 9 to 10 bits of precision for a memristor weight. Finally, the largest implemented crossbar is a 128×64 1T1M array [111] [19] [51] [112]. These performance parameters can be useful for optimizing the circuitry that interfaces a memristor crossbar with the other components of an intelligent system. As outlined in previous sections, choosing the right memristor crossbar for a system will depend upon the system's constraints and goals. If the system is meant to execute a large model with many parameters requiring high precision, a 1T1M array would be recommended, while 1S1M or memristor-only arrays can be beneficial in severely size-constrained applications [28].

B. Design Integration

As memristor technology has developed over recent years, so has our ability to integrate crossbars into powerful architectures. While crossbars are limited in size by parasitics and practicality, IBM has developed architectures comprised of up to 1 million memristive synapses [113]. Another recent work [114] demonstrates a memristor crossbar-based system for CNNs fully implemented in hardware. Eight chips (130 nm CMOS technology) were integrated on a printed circuit board (PCB), each of which includes a 128×16 1T1M array, decoding logic, drivers, and multiplexers for operating the arrays in memory or compute mode, 8-bit ADCs, and shift-and-add circuitry. Also included in the PCB were DRAM chips and an ARM microcontroller. The memristors were multicell and capable of 32 different states from 2-20μS, of which 15 states were used for the demonstration. A five-layer CNN was trained for MNIST in Python with an accuracy of 97.99 percent, or 96.92 percent after quantization. When the memristors were programmed and the system was tested, the hardware implementation achieved 95.07 percent accuracy. To account for device variation and programming noise, in-situ training of the fully-connected layer is completed for an epoch, improving the accuracy to 96.19 percent. The resulting computational efficiency of the full system is estimated to be 11,014 GOPS per W. It's worth noting that the ADCs are the largest and most power-hungry blocks of the chip by far.

Another system fully integrates a 54×108 (memristor only) crossbar with on-chip 4-bit DACs, 13-bit ADCs, interface circuits, and an OpenRISC processor in 180 nm CMOS technology [115]. The system is able to achieve an accuracy of 94.6 percent on a breast cancer screening data-set using a two-layer network: the first layer implements principal component analysis (PCA) using online learning [116] on one portion of the crossbar; the second layer performs a single-layer perceptron (SLP) on a separate portion of the crossbar. It is estimated that the system sports a computational efficiency of 187.62 GOPS per W at 180 nm technology, but the authors claim that 1.37 TOPS per W (1,370 GOPS per W) can be achieved with 40 nm technology, not including other potential improvements from reducing the ADC precision or selecting a more efficient processor core.

Slightly more recent works integrate memristor crossbar arrays with CMOS technology directly on-chip to minimize losses due to communication over PCB [102] [103]. These works provide 11.91 TOPS per W for 8-bit input, 8-bit weight, and 14-bit output MACs at 22 nm, and 36.4 TOPS per W for 1-bit input, 1-bit weight MACs at 40 nm, respectively.

VI. Memristor In-Situ Learning Networks

Memristor-based neuromorphic learning can be divided into ex-situ and in-situ learning styles. Ex-situ learning refers to the training of a neural network in software, after which the final weight is written into memristor memory [117]. In contrast, in-situ learning means that every synapse or weight updated is reflected by the updating of memristor memory throughout the entire learning process. That is to say, the conductance of the memristor synapses is updated in real-time [118]. Meanwhile, the calculation of synaptic weight updates can be accomplished through external circuits, sometimes using the memristors themselves in the process. In this way, in-situ learning can adapt to the characteristics and defects of hardware devices and has better learning stability than ex-situ learning. Nevertheless, in-situ learning needs to continually adjust or write memristor conductance throughout the learning process, which can be complicated and difficult to implement.

To avoid the influence of sneak path (i.e., the undesired current flowing through unselected cells) and quickly update the conductances of all memristors, 1T1M structure is preferred in multiple designs. For instance, Dong *et al.* [118] proposed a memristor-based multilayer perceptron (MLP) using the 1T1M synaptic structure and exploited in-situ learning by applying compact extreme learning machine to bimodal memristive synapses. Besides, Ryu and Kim [119] constructed long short-term memory (LSTM) networks via 1T1M synapses to achieve in-situ learning. In fact, for in-situ learning, accurately adjusting the memristor conductance according to the weight value is also a challenge. A naïve solution is to utilize linear models to calculate the duration of voltage pulse corresponding to the variation of conductances, whereas this approach cannot be applied to the actual nonlinear memristor. For 1T1M synaptic, the programming of conductance can be controlled by the voltage-gated amplitude of the transistor.

Here, an example of a memristor-based LSTM circuit with in-situ learning ability is presented [120]. In this example, a type of memristor-based LSTM circuit with in-situ learning ability [120] is designed, which consists of memristor-based LSTM cells, a memristor-based fully-connected layer, and a winner-take-all (WTA) module. To fulfill the LSTM functionality, memristor-based LSTM circuits are equipped with feedforward and recurrent structures and have the ability to process time-series data, and then in-situ learning is operated by updating the memductances in the memristor-based LSTM circuit in parallel.

Recently, some explorations are to design HTM spatial pooling with memristor crossbar circuits [121]. The characteristics of this HTM spatial pooling with memristor crossbar circuits are as follows: (1) It can accomplish the functions of strength and state of synaptic connections at the same time; (2) Local region suppression with fixed radius is used to replace global suppression. For the input with topological structure, local region suppression is more effective than global suppression; (3) WTA circuit is applied to perform suppression operation; (4) By developing in-situ updating strategy that can renovate the strength and state of synaptic connections in parallel, the learning process is completely realized by the designed circuit, then the training of spatial pooling is accelerated. Based on on-device training, the HTM spatial pooling with memristor crossbar circuits is trained by competitive Hebbian learning rule. Although it is not as accurate as of the back propagation algorithm, it has the advantages of fast execution, simple circuit implementation, and low power consumption. Moreover, HTM spatial pooling with memristor crossbar circuits solves the problems of full implementation and insufficient parallelism for spatial pooling synapse that previous spatial pooling circuits suffer bottleneck.

VII. Challenges in Memristor Studies

In recent years, more and more researchers have begun to study the properties of memristors and establish more realistic memristor models. In order to explore the properties of memristors, different mathematical models of memristors are compared [122]. For example, the switching properties of TiO_2 based memristors are described in more detail in [123]. Simultaneously, different SPICE models of memristors and their applications in circuit design are investigated [124], [125]. One significant challenge is designing circuit models that can meet the requirements of existing memristor devices. Therefore, new memristor models are constantly established, including magnetic flux-controlled memristor models [126], threshold adaptive memristor (TeAM) models [127], and voltage-controlled threshold adaptive memristor (VTeAM) models [128]. However, each existing memristor model still cannot fully satisfy all demands of memristor circuit simulations. For example: (1) Magnetic flux-controlled memristor models lack threshold characteristics; (2) TeAM models have a current threshold, while most real memristor devices are activated by a voltage threshold; (3) VTeAM models have a voltage threshold, but the resistance of the memristor changing with voltage cannot be well simulated. Hence, there is an urgent need to propose circuit models that reflect the intrinsic characteristics of memristors rather than ideal prototypes, which can more accurately reveal the voltage-current traits of real memristors. Thus, these models would be able to emulate a variety of real memristor devices with different materials and could be applied to memristor neural network circuits.

Using memristor synapse arrays to design cellular neural networks has been proven to accelerate neural network operations [129]. Moreover, memristors are used to construct other neural network circuits, such as RBF neural networks [130], Hopfield neural networks [131], echo state networks [132], spiking neural networks [133], convolution neural networks [134], etc. However, the learning algorithms in the above networks all need computer-aided calculations, and directly implementing these neural network algorithms in hardware is a bottleneck on performance. Implementing neural networks with memristor circuits fundamentally changes how these algorithms can be executed. Subsequently, many scholars have carried out extensive research in this direction, including backpropagation-based algorithms, WTA algorithms, random adjustment algorithms, and so forth. However, the existing research still lacks success in circuit-level implementations of memristor neural networks for realizing pattern recognition functions with low error in hardware.

Due to the rapid development of brain-inspired computing, memristor-based neural network processing-in-memory architectures are outperforming traditional von-Neumann computer architectures [135]. However, memristor-based in-memory computing hardware is plagued by oversized peripheral circuit area and excessively low functional unit utilization. Possible solutions to these issues are aggregating peripheral circuits to realize functional units and proposing data mapping strategies to further improve the utilization of functional units and reduce the data transmission between memristor cubes.

For potential applications of memristor-based neuromorphic networks, scholars have studied correlative pattern recognition implemented in hardware [136]. However, there have been few previous studies that determined how to combine memristor-based neuromorphic networks with memristor data storage to achieve more complex functional applications. In addition, it is also very important for the design of peripheral control circuits and the optimization of weight adjustment processes in memristor-based neuromorphic networks [137]. Adopting new algorithms to create a memristor-based neuromorphic network with fewer memristors and smaller errors, as well as the ability to complete the complex pattern recognition functions with faster speed and higher accuracy, is worthy of further investigation.

In order to accelerate the commercialization of memristors, it is critical to investigate how to reduce the failure rates, extend the service life and minimize interference. Firstly, memristor reliability is lower than that of traditional CMOS circuits, and the failure rate of a single memristor unit is significantly higher than that of a CMOS unit. Secondly, due to the integration of different types of chips, the fabrication of CMOS-memristor hybrid integrated circuits adds many additional steps and introduces additional defect risks. Moreover, each chip layer contains a large number of functional modules. If any one of the modules, or their interconnection, or the interconnection between each layer encounters manufacturing defects, the entire chip may fail, and the product failure

rate is high. Furthermore, in a single device, multiple read operations may alter the state of the device, and detecting and avoiding this phenomenon is very important to the reliability of data. Finally, in CMOS-memristor hybrid circuits, because multiple memristor units are connected to a single nanowire, it is easy to interfere with adjacent units when operating on a single memristor unit.

VIII. Future Work

From current research trends in memristor technology, the following issues are worth particular emphasis in future research:

(1) By analyzing memristor devices made of different materials, more stable and reliable memristor materials and preparation methods should be pursued. The key to solving the problem of low reliability in many applications based on memristors is to improve the reliability of a single memristor. Improving the manufacturing process of memristors or using new manufacturing materials to improve the reliability of single memristor units will play a decisive role in the applications of memristors.

(2) Whether digital or analog computation, the fluctuation of memristor device properties across different cycles or in different devices can have a large impact on calculation results. For example, the fluctuations of SET threshold voltages for different memristor devices will lead to discrepancies in operation. Especially for accurate scientific calculation, the conformance requirement of memristor devices is relatively higher. The use of verification methods or redundancy design can improve tolerance of these errors to a certain extent, but it will incur additional energy consumption and delay, weakening the inherent advantages of in-memory computing based on memristors. Therefore, the conformance of memristor devices is an important problem.

(3) The reconfigurable memristor system based on memristor neurons should be optimized and refined. Each memristor subarray or processing unit can be considered as a functional module. The memristor system design should consider a variety of network structures and provide adaptable adjustment according to the requirement. Besides, the information transmission and connection should also be flexible.

(4) Learning algorithms that are suitable for implementation by memristor circuits should be explored for multilayer neural networks, as well as designs for the corresponding online learning and adjustment circuits.

(5) By designing new learning algorithms which can make full use of memristor characteristics, so that network learning and training speeds are faster, errors are mitigated, and noise tolerance is improved. One path forward is to build memristor-based biological network circuits to simulate memory functions of the human brain. Another path forward is to establish memristor-based integrated network circuits to realize non-von Neumann computational paradigms so as to innovate a new generation of intelligent computer architectures.

References

[1] L. O. Chua, "Memristor—The missing circuit element," *IEEE Trans. Circuit Theory*, vol. CT-18, no. 5, pp. 507–519, Sep. 1971.

[2] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[3] C.-W. Hsu *et al.*, "Self-rectifying bipolar TaO_x/TiO_2 RRAM with superior endurance over 10^{12} cycles for 3D high-density storage-class memory," in *Proc. Symp. VLSI Technol.*, 2013, pp. T166–T167.

[4] X. Yang, B. Yan, H. Li, and Y. Chen, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–9.

[5] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1864–1878, Oct. 2014.

[6] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 541–552.

[7] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 27–39, Jun. 2016.

[8] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.

[9] D. P. Sahu, P. Jetty, and S. N. Jammalamadaka, "Graphene oxide based synaptic memristor device for neuromorphic computing," *Nanotechnology*, vol. 32, no. 15, Apr. 2021, Art. no. 155701.

[10] L. Chen, W. Zhou, C. Li, and J. Huang, "Forgetting memristors and memristor bridge synapses with long- and short-term memories," *Neurocomputing*, vol. 456, pp. 126–135, Oct. 2021.

[11] P. Yao *et al.*, "Face classification using electronic synapses," *Nature Commun.*, vol. 8, no. 1, pp. 1–8, May 2017.

[12] S. K. Dubey, A. Reddy, R. Patel, M. Abz, A. Srinivasulu, and A. Islam, "Architecture of resistive RAM with write driver," *Solid State Electron. Lett.*, vol. 2, pp. 10–22, Dec. 2020.

[13] M. M. Majdabadi, S. B. Shokouhi, and S.-B. Ko, "Efficient hybrid CMOS/memristor implementation of bidirectional associative memory using passive weight array," *Microelectron. J.*, vol. 98, Apr. 2020, Art. no. 104725.

[14] R. Ai, N. Zu, and R. Li, "From gradual change to abrupt change in Ni-Al layered double hydroxide memristor by adsorbed small molecule oxadiazole," *Sens. Actuators A, Phys.*, vol. 323, Jun. 2021, Art. no. 112671.

[15] M. Hu, H. Li, Q. Wu, G. S. Rose, and Y. Chen, "Memristor crossbar based hardware realization of BSB recall function," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–7.

[16] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015.

[17] F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov, "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," *Nature Commun.*, vol. 9, no. 1, pp. 1–7, Dec. 2018.

[18] H. Kim, M. R. Mahmoodi, H. Nili, and D. B. Strukov, "4K-memristor analog-grade passive crossbar circuit," *Nature Commun.*, vol. 12, no. 1, pp. 1–11, Dec. 2021.

[19] C. Li *et al.*, "Efficient and self-adaptive *in-situ* learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 1, pp. 1–8, Dec. 2018.

[20] C.-X. Xue *et al.*, "A 22 nm 2 Mb ReRAM compute-in-memory macro with 121–28 TOPS/W for multibit MAC computing for tiny AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 244–246.

[21] J.-J. Huang, Y.-M. Tseng, W.-C. Luo, C.-W. Hsu, and T.-H. Hou, "One selector-one resistor (1S1R) crossbar array for high-density flexible memory applications," in *IEDM Tech. Dig.*, Dec. 2011, pp. 7–31.

[22] Y. Deng *et al.*, "RRAM crossbar array with cell selection device: A device and circuit interaction study," *IEEE Trans. Electron Device*, vol. 60, no. 2, pp. 719–726, Feb. 2013.

[23] Y. Huang *et al.*, "A TaO_x based threshold switching selector for the RRAM crossbar array memory," in *Proc. 12th Annu. Non-Volatile Memory Technol. Symp.*, Oct. 2012, pp. 85–87.

[24] V. S. S. Srinivasan *et al.*, "Punchthrough-diode-based bipolar RRAM selector by Si epitaxy," *IEEE Electron Device Lett.*, vol. 33, no. 10, pp. 1396–1398, Oct. 2012.

[25] A. Kawahara *et al.*, "An 8 Mb multi-layered cross-point ReRAM macro with 443 MB/s write throughput," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 178–185, Jan. 2013.

[26] L. Zhao, L. Jiang, Y. Zhang, N. Xiao, and J. Yang, "Constructing fast and energy efficient 1TnR based ReRAM crossbar memory," in *Proc. 18th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2017, pp. 58–64.

[27] Z. Wang, S. Ambrogio, S. Balatti, and D. Ielmini, "A 2-transistor/1-resistor artificial synapse capable of communication and stochastic learning in neuromorphic systems," *Frontiers Neurosci.*, vol. 8, p. 438, Jan. 2015.

[28] Q. Xia and J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Mater.*, vol. 18, no. 4, pp. 309–323, Apr. 2019.

[29] C. Liu and H. Li, "A weighted sensing scheme for ReRAM-based cross-point memory array," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2014, pp. 65–70.

[30] M. Zackriya, H. M. Kittur, and A. Chin, "A novel read scheme for large size one-resistor resistive random access memory array," *Sci. Rep.*, vol. 7, no. 1, pp. 1–7, Mar. 2017.

[31] L. Shi, G. Zheng, B. Tian, B. Dkhil, and C. Duan, "Research progress on solutions to the sneak path issue in memristor crossbar arrays," *Nanosc. Adv.*, vol. 2, no. 5, pp. 1811–1827, 2020.

[32] V. A. Demin, I. A. Surazhevsky, A. V. Emelyanov, P. K. Kashkarov, and M. V. Kovalchuk, "Sneak, discharge, and leakage current issues in a high-dimensional 1T1M memristive crossbar," *J. Comput. Electron.*, vol. 19, no. 2, pp. 565–575, Jun. 2020.

[33] S. Kvatinsky *et al.*, "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[34] S. Gupta, M. Imani, and T. Rosing, "FELIX: Fast and energy-efficient logic in memory," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2018, pp. 1–7.

[35] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic design within memristive memories using memristor-aided loGIC (MAGIC)," *IEEE Trans. Nanotechnol.*, vol. 15, no. 4, pp. 635–650, Jul. 2016.

[36] A. Haj-Ali, R. Ben-Hur, N. Wald, and S. Kvatinsky, "Efficient algorithms for in-memory fixed point multiplication using MAGIC," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[37] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "FloatPIM: In-memory acceleration of deep neural network training with high precision," in *Proc. ACM/IEEE 46th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2019, pp. 802–815.

[38] X. Liu *et al.*, "RENO: A high-efficient reconfigurable neuromorphic computing accelerator design," in *Proc. 52nd Annu. Design Automat. Conf.*, Jun. 2015, pp. 1–6.

[39] C. Liu *et al.*, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 110–115.

[40] M. F. Chang *et al.*, "A high-speed 7.2-ns read-write random access 4-Mb embedded resistive RAM (ReRAM) macro using process-variation-tolerant current-mode read schemes," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 878–891, Mar. 2013.

[41] A. Prakash, J. Park, J. Song, J. Woo, E.-J. Cha, and H. Hwang, "Demonstration of low power 3-bit multilevel cell characteristics in a TaO_x-based RRAM by stack engineering," *IEEE Electron Device Lett.*, vol. 36, no. 1, pp. 32–34, Jan. 2015.

[42] Q. Zheng *et al.*, "MobiLattice: A depth-wise DCNN accelerator with hybrid digital/analog nonvolatile processing-in-memory block," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2020, pp. 1–9.

[43] P. Lin *et al.*, "Three-dimensional memristor circuits as complex neural networks," *Nature Electron.*, vol. 3, pp. 225–232, Apr. 2020.

[44] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[45] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[46] C. Liu *et al.*, "A spiking neuromorphic design with resistive crossbar," in *Proc. 52nd Annu. Design Automat. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[47] B. Yan *et al.*, "RRAM-based spiking nonvolatile computing-in-memory processing engine with precision-configurable *in situ* nonlinear activation," in *Proc. Symp. VLSI Technol.*, Jun. 2019, pp. T86–T87.

[48] H. Jiang *et al.*, "Pulse-width modulation based dot-product engine for neuromorphic computing system using memristor crossbar array," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–4.

[49] Y. Long, E. M. Jung, J. Kung, and S. Mukhopadhyay, "ReRAM crossbar based recurrent neural network for human activity detection," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 939–946.

[50] Y. Long, T. Na, and S. Mukhopadhyay, "ReRAM-based processing-in-memory architecture for recurrent neural network acceleration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 12, pp. 2781–2794, Dec. 2018.

[51] C. Li *et al.*, "Long short-term memory networks in memristor crossbar arrays," *Nature Mach. Intell.*, vol. 1, pp. 49–57, Jan. 2019.

[52] K. Adam, K. Smagulova, and A. P. James, "Memristive LSTM network hardware architecture for time-series predictive modeling problems," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Oct. 2018, pp. 459–462.

[53] K. Smagulova, K. Adam, O. Krestinskaya, and A. P. James, "Design of CMOS-memristor circuits for LSTM architecture," in *Proc. IEEE Int. Conf. Electron Devices Solid State Circuits (EDSSC)*, Jun. 2018, pp. 1–2.

[54] K. Smagulova, O. Krestinskaya, and A. P. James, "A memristor-based long short term memory circuit," *Analog Integr. Circuits Signal Process.*, vol. 95, no. 3, pp. 467–472, 2018.

[55] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on RRAM," in *Proc. 22nd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2017, pp. 782–787.

[56] X. Sun, S. Yin, X. Peng, R. Liu, J.-S. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1423–1428.

[57] L. Song, Y. Wu, X. Qian, H. Li, and Y. Chen, "ReBNN: *In-situ* acceleration of binarized neural networks in ReRAM using complementary resistive cell," *CCF Trans. High Perform. Comput.*, vol. 1, nos. 3–4, pp. 196–208, Dec. 2019.

[58] B. Kim, E. Hanson, and H. Li, "An efficient 3D ReRAM convolution processor design for binarized weight networks," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 5, pp. 1600–1604, May 2021.

[59] Z. Wang *et al.*, "Reinforcement learning with analogue memristor arrays," *Nature Electron.*, vol. 2, no. 3, pp. 115–124, Mar. 2019.

[60] F. Chen, L. Song, and Y. Chen, "ReGAN: A pipelined ReRAM-based accelerator for generative adversarial networks," in *Proc. 23rd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2018, pp. 178–183.

[61] F. Chen, L. Song, H. H. Li, and Y. Chen, "ZARA: A novel zero-free dataflow accelerator for generative adversarial networks in 3D ReRAM," in *Proc. 56th Annu. Design Automat. Conf.*, Jun. 2019, pp. 1–6.

[62] Z. Fan, Z. Li, B. Li, Y. Chen, and H. Li, "RED: A ReRAM-based deconvolution accelerator," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1763–1768.

[63] W. Huangfu, S. Li, X. Hu, and Y. Xie, "RADAR: A 3D-ReRAM based DNA alignment accelerator architecture," in *Proc. 55th ACM/ESDA/IEEE Design Automat. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[64] S. Gupta, M. Imani, B. Khaleghi, V. Kumar, and T. Rosing, "RAPID: A ReRAM processing in-memory architecture for DNA sequence alignment," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[65] F. Zokaee, M. Zhang, and L. Jiang, "FindeR: Accelerating FM-index-based exact pattern matching in genomic sequences through ReRAM technology," in *Proc. 28th Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, Sep. 2019, pp. 284–295.

[66] F. Chen, L. Song, H. Li, and Y. Chen, "PARC: A processing-in-CAM architecture for genomic long read pairwise alignment using ReRAM," in *Proc. 25th Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2020, pp. 175–180.

[67] L. Song, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "GraphR: Accelerating graph processing using ReRAM," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 531–543.

[68] G. Dai, T. Huang, Y. Wang, H. Yang, and J. Wawrzynek, "GraphSAR: A sparsity-aware processing-in-memory architecture for large-scale graph processing on ReRAMs," in *Proc. 24th Asia South Pacific Design Automat. Conf.*, Jan. 2019, pp. 120–126.

[69] M. Zhou, M. Imani, S. Gupta, Y. Kim, and T. Rosing, "GRAM: Graph processing in a ReRAM-based computational memory," in *Proc. IEEE Asia South Pacific Design Automat. Conf.*, Jan. 2019, pp. 1–6.

[70] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, "Enabling scientific computing on memristive accelerators," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 367–382.

[71] M. A. Zidan *et al.*, "A general memristor-based partial differential equation solver," *Nature Electron.*, vol. 1, no. 7, pp. 411–420, 2018.

[72] L. Song, F. Chen, X. Qian, H. Li, and Y. Chen, "Low-cost floating-point processing in ReRAM for scientific computing," 2020, *arXiv:2011.03190*.

[73] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[74] L. Xia *et al.*, "MNSIM: Simulation platform for memristor-based neuromorphic computing system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 1009–1022, May 2018.

[75] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.

[76] S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, "RxNN: A framework for evaluating deep neural networks on resistive crossbars," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 2, pp. 326–338, Feb. 2021.

[77] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th Annu. Design Automat. Conf.*, Jun. 2019, pp. 1–6.

[78] M. J. Rasch *et al.*, "A flexible and fast PyTorch toolkit for simulating training and inference on analog crossbar arrays," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2021, pp. 1–4.

[79] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, "Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems," in *Proc. 54th Annu. Design Automat. Conf.*, Jun. 2017, pp. 1–6.

[80] B. Li, B. Yan, C. Liu, and H. Li, "Build reliable and efficient neuromorphic design with memristor technology," in *Proc. 24th Asia South Pacific Design Automat. Conf.*, Jan. 2019, pp. 224–229.

[81] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *Proc. 54th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2017, pp. 1–6.

[82] C.-F. Lee, H.-J. Lin, C.-W. Lien, Y.-D. Chih, and J. Chang, "A 1.4 Mb 40-nm embedded ReRAM macro with 0.07 μm^2 bit cell, 2.7 mA/100 MHz low-power read and hybrid write verify for high endurance application," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2017, pp. 9–12.

[83] B. Feinberg, S. Wang, and E. Ipek, "Making memristive neural network accelerators reliable," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 52–65.

[84] Q. Lou, T. Gao, P. Faley, M. Niemier, X. S. Hu, and S. Joshi, "Embedding error correction into crossbars for reliable matrix vector multiplication using emerging devices," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, Aug. 2020, pp. 139–144.

[85] I. Yeo, M. Chu, S.-G. Gi, H. Hwang, and B.-G. Lee, "Stuck-at-fault tolerant schemes for memristor crossbar array-based neural networks," *IEEE Trans. Electron Devices*, vol. 66, no. 7, pp. 2937–2945, Jul. 2019.

[86] Z. Wei *et al.*, "Demonstration of high-density ReRAM ensuring 10-year retention at 85C based on a newly developed reliability model," in *IEDM Tech. Dig.*, Dec. 2011, pp. 4–31.

[87] Z. Wei *et al.*, "Retention model for high-density ReRAM," in *Proc. 4th IEEE Int. Memory Workshop*, May 2012, pp. 1–4.

[88] Y. Y. Chen *et al.*, "Postcycling LRS retention analysis in HfO_2/Hf RRAM 1T1R device," *IEEE Electron Device Lett.*, vol. 34, no. 5, pp. 626–628, May 2013.

[89] S. Fukuyama, A. Hayakawa, R. Yasuhara, S. Matsuda, H. Kinoshita, and K. Takeuchi, "Comprehensive analysis of data-retention and endurance trade-off of 40 nm TaO_x-based ReRAM," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Mar. 2019, pp. 1–6.

[90] M.-J. Lee *et al.*, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures," *Nature Mater.*, vol. 10, no. 8, pp. 625–630, Aug. 2011.

[91] W. Chen *et al.*, "Switching characteristics of W/Zr/HfO_2/TiN ReRAM devices for multi-level cell non-volatile memory applications," *Semicond. Sci. Technol.*, vol. 30, no. 7, Jul. 2015, Art. no. 075002.

[92] U. Chand, C.-Y. Huang, J.-H. Jieng, W.-Y. Jang, C.-H. Lin, and T.-Y. Tseng, "Suppression of endurance degradation by utilizing oxygen plasma treatment in HfO_2 resistive switching memory," *Appl. Phys. Lett.*, vol. 106, no. 15, Apr. 2015, Art. no. 153502.

[93] Q. Wu, W. Banerjee, J. Cao, Z. Ji, L. Li, and M. Liu, "Improvement of durability and switching speed by incorporating nanocrystals in the HfO_x based resistive random access memory devices," *Appl. Phys. Lett.*, vol. 113, no. 2, Jul. 2018, Art. no. 023105.

[94] W. Wen, Y. Zhang, and J. Yang, "ReNEW: Enhancing lifetime for ReRAM crossbar based neural network accelerators," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, Nov. 2019, pp. 487–496.

[95] G. L. Zhang *et al.*, "Reliable and robust RRAM-based neuromorphic computing," in *Proc. Great Lakes Symp. VLSI*, Sep. 2020, pp. 33–38.

[96] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, G. De Micheli, and R. Drechsler, "Endurance management for resistive logic-in-memory computing architectures," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1092–1097.

[97] B. Liu *et al.*, "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 63–70.

[98] H. Li *et al.*, "Variation-aware, reliability-emphasized design and optimization of RRAM using SPICE model," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, 2015, pp. 1425–1430.

[99] M. Zangeneh and A. Joshi, "Design and optimization of nonvolatile multibit 1T1R resistive RAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 8, pp. 1815–1828, Aug. 2014.

[100] X. Yang *et al.*, "Multi-objective optimization of ReRAM crossbars for robust DNN inferencing under stochastic noise," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2021, pp. 1–9.

[101] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1769–1774.

[102] C.-X. Xue *et al.*, "A 22 nm 4 Mb 8b-precision ReRAM computing-in-memory macro with 11.91 to 195.7 TOPS/W for tiny AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 245–247.

[103] W. Li, X. Sun, H. Jiang, S. Huang, and S. Yu, "A 40 nm RRAM compute-in-memory macro featuring on-chip write-verify and offset-cancelling ADC references," in *Proc. IEEE 47th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2021, pp. 79–82.

[104] B. Yan, J. Yang, Q. Wu, Y. Chen, and H. Li, "A closed-loop design to enhance weight stability of memristor based neural network chips," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 541–548.

[105] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 19–24.

[106] J. Johnson, "Thermal agitation of electricity in conductors," *Phys. Rev.*, vol. 32, no. 1, p. 97, 1928.

[107] H. Shin, M. Kang, and L.-S. Kim, "A thermal-aware optimization framework for ReRAM-based deep neural network acceleration," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–9.

[108] S. Pi, C. Li, H. Jiang, W. Xia, H. Xin, J. J. Yang, and Q. Xia, "Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension," *Nature Nanotechnol.*, vol. 14, no. 1, pp. 35–39, Nov. 2018.

[109] B. J. Choi *et al.*, "High-speed and low-energy nitride memristors," *Adv. Funct. Mater.*, vol. 26, no. 29, pp. 5290–5296, 2016.

[110] Y. Van de Burgt *et al.*, "A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing," *Nature Mater.*, vol. 16, no. 4, pp. 414–418, Apr. 2017.

[111] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electron.*, vol. 1, no. 1, pp. 52–59, 2018.

[112] M. Hu *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater.*, vol. 30, no. 9, 2018, Art. no. 1705914.

[113] S. Wozniak, A. Pantazi, S. Sidler, N. Papandreou, Y. Leblebici, and E. Eleftheriou, "Neuromorphic architecture with 1M memristive synapses for detection of weakly correlated inputs," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 11, pp. 1342–1346, Nov. 2017.

[114] P. Yao *et al.* "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[115] F. Cai *et al.*, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electron.*, vol. 2, no. 7, pp. 290–299, Jul. 2019.

[116] S. Choi, J. H. Shin, J. Lee, P. Sheridan, and W. D. Lu, "Experimental demonstration of feature extraction and dimensionality reduction using memristor networks," *Nano Lett.*, vol. 17, no. 5, pp. 3113–3118, Apr. 2017.

[117] C. Yakopcic, R. Hasan, and T. M. Taha, "Memristor based neuromorphic circuit for *ex-situ* training of multi-layer neural network algorithms," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–7.

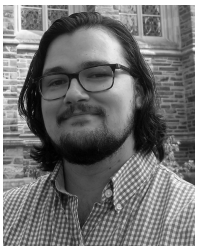
[118] Z. Dong, C. S. Lai, Z. Zhang, D. Qi, M. Gao, and S. Duan, "Neuromorphic extreme learning machines with bimodal memristive synapses," *Neurocomputing*, vol. 453, pp. 38–49, Sep. 2021.

[119] H. Ryu and S. Kim, "Implementation of a reservoir computing system using the short-term effects of Pt/HfO_2/TaO_x/TiN memristors with self-rectification," *Chaos, Solitons Fractals*, vol. 150, Sep. 2021, Art. no. 111223.

[120] M. S. Feali, "Using volatile/non-volatile memristor for emulating the short- and long-term adaptation behavior of the biological neurons," *Neurocomputing*, vol. 465, pp. 157–166, Nov. 2021.
[121] X. Liu, Y. Huang, Z. Zeng, and D. C. Wunsch, "Memristor-based HTM spatial pooler with on-device learning for pattern recognition," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 3, pp. 1901–1915, Mar. 2022.
[122] A. Ascoli, F. Corinto, V. Senger, and R. Tetzlaff, "Memristor model comparison," *IEEE Circuits Syst. Mag.*, vol. 13, no. 2, pp. 89–105, 2nd Quart., 2013.
[123] J. L. Tedesco, L. Stephey, M. Hernandez-Mora, C. A. Richter, and N. Gergel-Hackett, "Switching mechanisms in flexible solution-processed TiO_2 memristors," *Nanotechnol.*, vol. 23, no. 30, 2012, Art. no. 305206.
[124] X. Zhang, Z. Huang, and J. Yu, "Memristor model for SPICE," *IEICE Trans. Electron.*, vol. E93-C, no. 3, pp. 355–360, 2010.
[125] S. Thomas and S. Prakash, "An accurate analytical memristor model for spice simulators," *IEICE Electron. Exp.*, vol. 15, no. 18, 2018, Art. no. 20180724.
[126] N. Raj, R. K. Ranjan, and F. Khateb, "Flux-controlled memristor emulator and its experimental results," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 4, pp. 1050–1061, Apr. 2020.
[127] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: Threshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013.
[128] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.
[129] Y. Li, J. Li, J. Li, S. Duan, L. Wang, and M. Guo, "A reconfigurable bidirectional associative memory network with memristor bridge," *Neurocomputing*, vol. 454, pp. 382–391, Sep. 2021.
[130] M. I. Khan, S. Ali, A. Al-Tamimi, A. Hassan, A. A. Ikram, and A. Bermak, "A robust architecture of physical unclonable function based on memristor crossbar array," *Microelectron. J.*, vol. 116, Oct. 2021, Art. no. 105238.
[131] C. Mo, C. Chengjie, B. Bocheng, and X. Quan, "Initial sensitive dynamics in memristor synapse-coupled Hopfield neural network," *Electron. J. Inf. Technol.*, vol. 42, no. 4, pp. 870–877, 2020.
[132] E. Wlazlak, P. Zawal, and K. Szaclowski, "Neuromorphic applications of a multivalued $[SnI_4\{(C_6H_5)_2SO\}_2]$ memristor incorporated in the echo state machine," *ACS Appl. Electron. Mater.*, vol. 2, no. 2, pp. 329–338, 2020.
[133] Z. Zhao *et al.*, "A memristor-based spiking neural network with high scalability and learning efficiency," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 5, pp. 931–935, May 2020.
[134] X. Zeng, S. Wen, Z. Zeng, and T. Huang, "Design of memristor-based image convolution calculation in convolutional neural network," *Neural Comput. Appl.*, vol. 30, no. 2, pp. 503–508, 2018.
[135] A. Haj-Ali, R. Ben-Hur, N. Wald, R. Ronen, and S. Kvatinsky, "Imaging: In-memory algorithms for image processing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4258–4271, Dec. 2018.
[136] S. N. Truong, K. Van Pham, W. Yang, and K.-S. Min, "Sequential memristor crossbar for neuromorphic pattern recognition," *IEEE Trans. Nanotechnol.*, vol. 15, no. 6, pp. 922–930, Nov. 2016.
[137] R. Ranjan *et al.*, "Integrated circuit with memristor emulator array and neuron circuits for biologically inspired neuromorphic pattern recognition," *J. Circuits, Syst. Comput.*, vol. 26, no. 11, Nov. 2017, Art. no. 1750183.

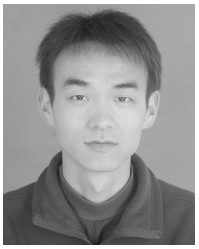
Xiaoxuan Yang (Student Member, IEEE) received the B.S. degree in electrical engineering from Tsinghua University in 2016 and the M.S. degree in electrical engineering from the University of California at Los Angeles (UCLA), Los Angeles, in 2018. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Duke University.

Her research interests include emerging nonvolatile memory technologies and hardware accelerators for deep learning applications.



Brady Taylor (Student Member, IEEE) received the B.S. degree in electrical engineering from Rice University, Houston, TX, USA, in 2019. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Duke University, supervised by Dr. Hai Helen Li.

His research focuses on mixed-signal VLSI implementations of neuromorphic systems and circuits that exploit the dynamics of emerging devices to develop biologically plausible learning rules.



Ailong Wu received the Ph.D. degree in systems analysis and integration from the Huazhong University of Science and Technology, Wuhan, China, in 2013.

He was a Post-Doctoral Research Fellow with Xi'an Jiaotong University, Xi'an, China, and also with Texas A&M University at Qatar, Doha, Qatar. He is currently a Professor with the College of Mathematics and Statistics, Hubei Normal University, Huangshi, China. His current research interests include nonlinear dynamics, hybrid systems, and associative memories.

Yiran Chen (Fellow, IEEE) received the B.S. and M.S. degrees from Tsinghua University in 1998 and 2001, respectively, and the Ph.D. degree from Purdue University in 2005. After five years in industry, he joined the University of Pittsburgh in 2010 as an Assistant Professor and then was promoted to an Associate Professor with tenure in 2014, holding the Bicentennial Alumni Faculty Fellow. He is currently a Professor with the Department of Electrical and Computer Engineering, Duke University. He is also currently serving as the Director for the NSF AI Institute for Edge Computing Leveraging Next-Generation Networks, Athens, and the NSF Industry–University Cooperative Research Center (IUCRC) for Alternative Sustainable and Intelligent Computing (ASIC); and the Co-Director of the Duke Center for Computational Evolutionary Intelligence (CEI). His group focuses on the research of new memory and storage systems, machine learning and neuromorphic computing, and mobile computing systems. He has published one book and about 500 technical publications, and has been granted 96 U.S. patents. He has served on the technical and organization committees of more than 60 international conferences. He is a fellow of the ACM. He received eight best paper awards, one best poster award, and 14 best paper nominations from the international conferences and workshops. He received many professional awards and is the Distinguished Lecturer of IEEE CEDA from 2018 to 2021. He currently serves as the Chair for ACM SIGDA. He has served as the associate editor for a dozen international academic transactions/journals. He is currently serving as the Editor-in-Chief for the *IEEE Circuits and Systems Magazine*.

Leon O. Chua (Life Fellow, IEEE) is widely known for his invention of the memristor. When not immersed in Science, he relaxes by searching for Wagner's leitmotifs, musing over Kandinsky's chaos, and contemplating Wittgenstein's inner thoughts. His research has been recognized through 17 honorary doctorates from major universities in Europe and Japan, and holds seven U.S. patents. He was elected as the Confrérie des Chevaliers du Tastevin in 2000. He was a Foreign Member of the European Academy of Sciences (Academia Europaea) in 1997 and the Hungarian Academy of Sciences in 2007. He was conferred numerous prestigious awards, including the first Kirchhoff Award, the Guggenheim Fellow, the 2019 EDS Celebrated Member Prize—The highest recognition of the IEEE Electron Devices Society, and the 2020 Julius Springer Prize in applied physics.