ADVANCED INTELLIGENT SYSTEMS Open Access
www.advintellsyst.com

Resistive Memory-Based In-Memory Computing: From Device and Large-Scale Integration System Perspectives

*Bonan Yan, Bing Li, Ximing Qiao, Cheng-Xin Xue, Meng-Fan Chang, Yiran Chen, and Hai (Helen) Li**

In-memory computing is a computing scheme that integrates data storage and arithmetic computation functions. Resistive random access memory (RRAM) arrays with innovative peripheral circuitry provide the capability of performing vector-matrix multiplication beyond the basic Boolean logic. With such a memory–computation duality, RRAM-based in-memory computing enables an efficient hardware solution for matrix-multiplication-dependent neural networks and related applications. Herein, the recent development of RRAM nanoscale devices and the parallel progress on circuit and microarchitecture layers are discussed. Well suited for analog synapse and neuron implementation, RRAM device properties and characteristics are emphasized herein. 3D-stackable RRAM and on-chip training are introduced in large-scale integration. The circuit design and system organization of RRAM-based in-memory computing are essential to breaking the von Neumann bottleneck. These outcomes illuminate the way for the large-scale implementation of ultra-low-power and dense neural network accelerators.

1. Introduction

The von Neumann architecture that has been widely adopted in modern computing systems was first presented in 1945 by von Neumann and others.[1] It is a basic digital computer architecture that separates the central processor unit (CPU) from the storage device, as shown in **Figure** 1a.[2] During data processing, it is necessary to transfer data between the memory and CPU through a transmission interface. As data volume increases, the high latency and energy consumption of data transmissions emerge as the new development bottleneck of the von Neumann architecture, namely, "von Neumann bottleneck" (Figure 1b).[3] System solutions, such as integrating multiple processing cores[3] and designing dedicated accelerators,[4] enhance the data processing speed while further aggravating the gap between computation and data transportation.

In-memory computing, also called as computing-in-memory (CIM) or process-in-memory (PIM), fuses memory modules and data processing units to reduce and even eliminate the frequent data transportation between memory and data processing units in modern computers. It is a promising way to build energy-efficient computing systems, echoing Backus's thoughts in 1978: "Surely there must be a less primitive way of making big changes in the store than by pushing vast numbers of words back and forth through the von Neumann bottleneck."[5] In fact, the concept of in-memory computing is not new. Back to a few decades ago, in-memory data management was a key problem for databases, especially in enterprise data warehouse, owning to the astoundingly huge amount of data.[6] The investigation on storage equipment to locally process data at that time remained a system-level study. In contrast, the scope of this paper focuses on the development of in-memory computing at the circuit and architecture levels, such as a physical-layer module that stores a part of operands and conducts computation with these operands. For example, when executing two vector-matrix multiplication (VMM) operations of $\boldsymbol{X}_{1\times N}\cdot\boldsymbol{Y}_{N\times M}$, an in-memory computing module can store the matrix $\boldsymbol{Y}_{N\times M}$ locally and stream only the vector $\boldsymbol{X}_{1\times N}$ (Figure 1c). Compared with a design based on conventional arithmetic units, such an in-memory module can avoid the transition of $\boldsymbol{Y}_{N\times M}$ from external memories. It is particularly useful when the matrix $\boldsymbol{Y}_{N\times M}$ involves in computation repeatedly. The scenario is very common in large-scale data processing applications, such as deep neural networks (DNNs) and graphic analytics. For example, matrix–matrix multiplication is the fundamental operation in DNNs, which can be further decomposed as multiply accumulate (MAC).[7] The weight matrices with massive reuse can greatly benefit from in-memory computing.

Nowadays, a large variety of nanoscale memory devices are investigated, offering diverse choices to develop large-scale, high-performance, and energy-efficient in-memory computing designs. It is commonly believed that the following types of random or sequential access memories have great potentials:

B. Yan, Dr. B. Li, X. Qiao, Prof. Y. Chen, Prof. H. Li
Department of Electrical and Computer Engineering
Duke University
100 Science Drive, Durham, NC 27708, USA
E-mail: hai.li@duke.edu

C.-X. Xue, Prof. M.-F. Chang
Department of Electrical Engineering
National Tsing Hua University
Delta Building No. 101, Section 2, Kuang-Fu Road, Hsinchu 30013, Taiwan

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.201900068.

DOI: 10.1002/aisy.201900068

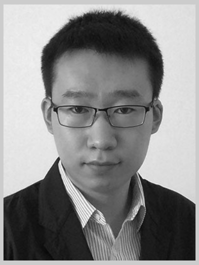
static random access memory (SRAM),[8–10] flash memory,[11,12] magnetic random access memory (MRAM),[13,14] racetrack memory,[15,16] phase change memory (PCM),[3,17–19] and resistive random access memory (RRAM).[20–22] All these memory types, except SRAM, are nonvolatile, i.e., a memory device maintains the stored data even without having power supply. SRAM features the fastest read/write speed (sub-nanosecond) and a much more mature fabrication process among the existing memory techniques.[8–10] Besides persistent data storage, nonvolatile memory technologies generally have a higher density, i.e., $<100\ F^2$ cell size, where F represents the technology feature size. NAND flash memory as a commercial nonvolatile memory technology supports sequential data accesses and has complicated techniques to maintain robustness and reliability. MRAM highlights fast write speed (in the same order as SRAM) and stochastic programming.[23–26] Racetrack memory is known for its extremely high density (20 nm-wide nanowire) and the sequential access along tracks.[27] PCM provides a linear conductance update characteristic.[18] RRAM offers versatility, including high resistivity (MΩ order of magnitude), the support of 3D integration, stochastic programming, and multilevel cell (up to 6 bits).[28–31]

The unique characteristics of different memory technologies make them useful in different application scenarios. Several recently published review articles summarized the exploitation of these memory technologies for in-memory computing from distinct aspects or layers. Xia and Yang described the physical mechanisms and fabrication of several state-of-the-art memristor devices and the principles to implement neural network hardware with them.[31] Jeong et al. reviewed how memristors progress from concepts to real-world devices that aim at energy-efficient computing paradigms.[32] Tsai et al. introduced the principal applications of analog memory devices in building deep learning accelerators, including prospective candidates of resistive, capacitive, and photonics devices.[30] Jeong and Hwang provided insights into the usage of nonvolatile memory materials to build machine learning computing hardware.[33] In this paper, we would like to broaden the sight beyond the focus solely on device layer or circuit layer, following the device/circuit/computer architecture/cross-layer codesign methodologies. Taking the RRAM device family as a representative technology, we review the latest progress of in-memory computing from the perspectives of state-of-the-art nanoscale devices and large-scale integration computing systems for artificial intelligence (AI) acceleration. We will start with the implementation of RRAM devices in weight matrices and in realizing activation function, followed by the circuit and architecture approaches of in-memory computing. The reliability issue as a new challenge in developing large-scale systems will also be discussed. In the end, we will conclude this paper.

2. RRAM Array for DNN Weighting Function

2.1. RRAM Basics and RRAM Array for Inference

A resistive memory (RRAM, a.k.a. memristor) device generally represents any two-terminal electronic device whose resistance value can be programmed by applying external voltage/current with an appropriate configuration.[34] A single RRAM device contains a resistive layer sandwiched by two electrodes. The resistive layer is typically transition metal oxides, such as HfO_x,[35–38] NbO_x,[39,40] TiO_x,[34,41,42] and TaO_x.[43,44] Other materials, such as $SrTiO_3$,[45] $PrCaMnO_3$,[45] and Ag:a-Si,[21] also show memristivity. According to the resistivity switching mechanisms, mainstream RRAM technologies can be classified into two categories:[29] filamentary RRAM that relies on the formation and dissolution of the conductive filaments or channels of the metal ions or oxygen vacancies in its insulating layer and interfacial RRAM that redistributes the oxygen vacancies at the heterogeneous interface to change the overall resistance. During programming, a "SET" operation increases the device conductance (or decreases the resistance), whereas a "RESET" operation decreases the conductance.


Bonan Yan received his bachelor's degree from Beihang University, China, and master's degree from the Department of Electrical and Computer Engineering, University of Pittsburgh, USA. He is pursuing his doctoral degree at Duke University, USA. His research interests include macromodeling and circuit design for emerging nonvolatile memories, brain-inspired computing systems, and hardware/software codesign for machine learning acceleration.

Meng-Fan Chang received his M.S. and Ph.D. degrees from Penn State University, USA, and National Chiao Tung University, Taiwan, respectively. He is a distinguished professor at National Tsing Hua University, Taiwan. Before 2006, he worked in the semiconductor industry over for 10 years. His research interests include circuit designs for volatile and nonvolatile memory, computing-in-computing, neuromorphic computing, circuit–device interaction beyond CMOS technologies, and software–hardware codesign for AI devices. He is an IEEE Fellow.

Hai Li received her bachelor's and master's degrees from Tsinghua University, China, and doctoral degree from the Department of Electrical and Computer Engineering, Purdue University, USA. She is currently the Clare Boothe Luce Associate Professor with the Department of Electrical and Computer Engineering at Duke University, USA. Her current research interests include hardware/software codesign for machine learning acceleration and security, brain-inspired computing systems, and memory architecture and optimization. Dr. Li is a distinguished speaker of ACM (2017–2020) and a distinguished lecture of the IEEE CAS society (2018–2019). Dr. Li is a fellow of IEEE and a distinguished member of ACM.

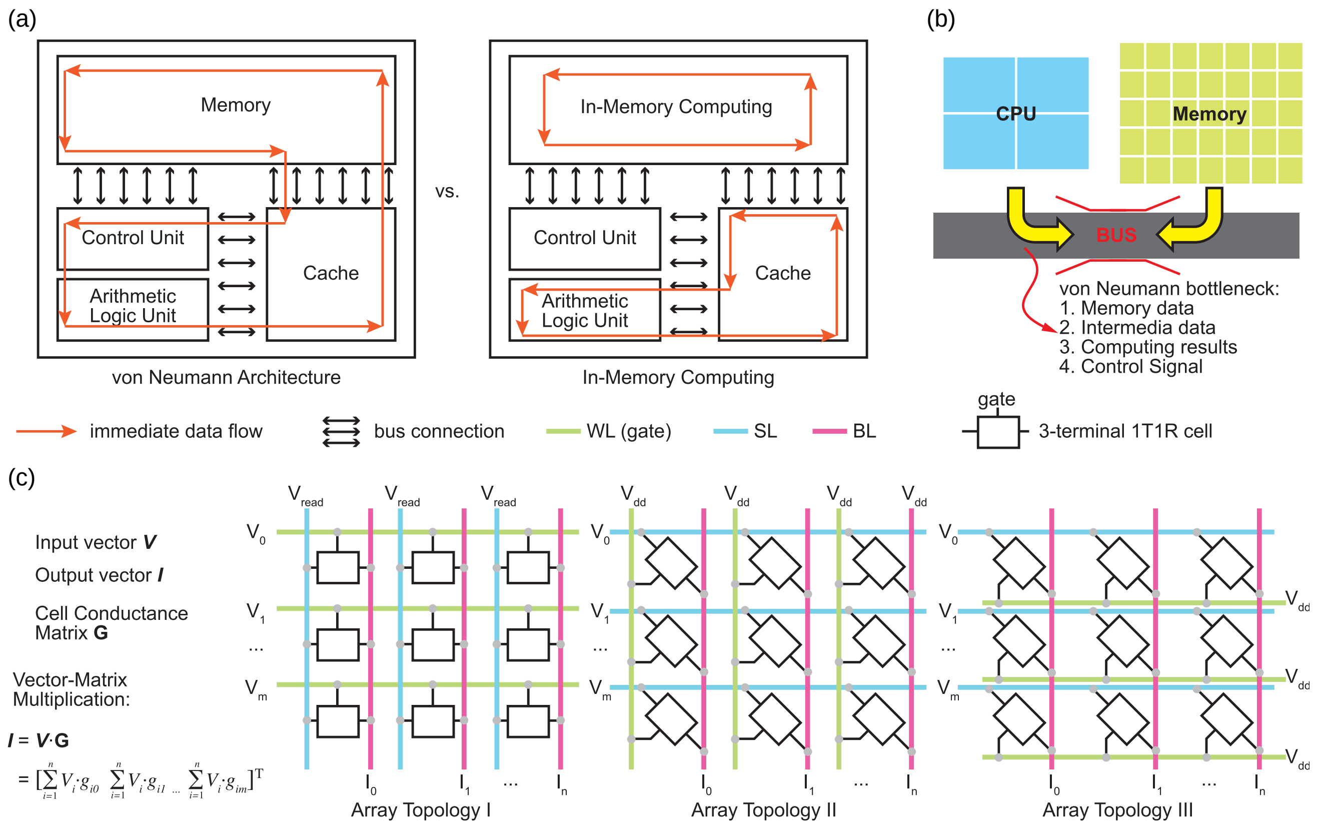


Figure 1. a) Schemes of von Neumann architecture versus in-memory computing.[2] b) Diagram of von Neumann bottleneck. c) Analog of RRAM array to VMM, where the synaptic weights are stored as a conductance matrix of the RRAM array.

In some types of RRAM technologies, often within the filamentary category, the "forming" operation is required.[46] It applies a much higher voltage than the programming one to generate initial filaments before normal use. According to the available stable resistance states, RRAM can be categorized into two types: analog RRAM denotes those devices whose resistances can be programmed to any value between the highest resistance state (HRS) and the lowest resistance state (LRS), whereas binary RRAM behaves as a normal memory device with a stable HRS and LRS. RRAM technology advances for its high data storage density as well as the compatibility with the complementary metal–oxide–semiconductor (CMOS) process. As early as in 2009, a 1 kb RRAM array in a one-transistor-one-RRAM (1T1R) cell structure was successfully integrated with CMOS read/write circuits.[20] In this design, the HfO_2-based RRAM with TiN electrodes is fabricated above the transistors. These devices achieved four separated resistance levels. Such a high storage density makes RRAM a great replacement of the conventional memory technology. The latest 3.6 Mb embedded binary RRAM array fabricated in a 22 nm low-power process is composed of 0.3 Mb subarrays together with read/write logic circuitry.[47]

Following the classical Boolean logic to construct computing automata, logic gate implementation with memory devices is the first attempt to exceed the function of memory and head for computation.[48–54] Binary inputs are stored into RRAM devices as the resistance (conductance) before performing computation.[55,56] With parallel or serial connection setups, sensing voltage is applied to the RRAM devices with data stored as resistances. The amplitude of output current represents the logic operation result. To be more specific, RRAM devices enable logic-in-memory to perform stateful logic operations. Xu et al. demonstrated a time-efficient implementation with dual-bit memristors for 12 different basic single-step logic operations, including TRUE, FALSE, NOT, AND, OR, and, more importantly, "material implication" (IMP).[50] A large-scale array of stateful logical RRAM is the aggregation of these basic logic gates similar to the transistor-based arithmetic units.[57] By weaving complex logics with stateful RRAM in a dataflow automation scheme, a fast and energy-efficient computer can be realized to execute heavy logical workload (see arithmetic operation).[53] The realization with stateful logic RRAMs often requires special routing techniques to attain logic gate functionality and high accuracy to resist IR drop. Thanks to nonvolatility, RRAM-based logic gate implementation targets at the normally off-computing systems, leading to dramatically the power consumption reduction of computing systems.[55,57]

The analog between memristors and biological synapse originated shortly after Hewlett Packard Labs discovered nanoscale RRAM devices as memristors in 2008 (**Figure** 2a).[34] Since then, there have been extensive studies on developing electric synapses with RRAM devices and exploiting them for DNN acceleration.[29,32,33] In 2012, Hu et al. described how to conduct VMM on an RRAM crossbar array based on Kirchhoff's law.[58] Furthermore, they analyzed the necessary design consideration

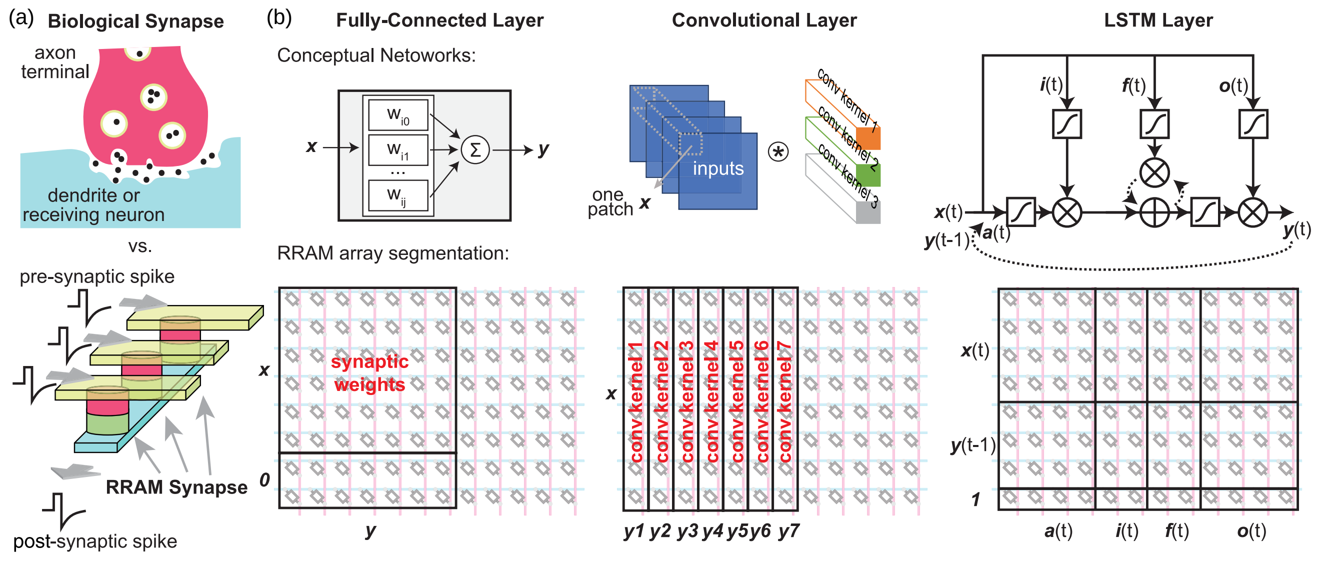


Figure 2. a) Biological synapse versus RRAM synapse.[5] b) Different segmentation schemes for fully connected layers (DNN), convolutional layers (CNN), and LSTM layers (LSTM network), all of which are converted into VMM using the RRAM array.

in real circuit implementation by taking a brain-state-in-a-box (BSB) computing model as an example. A spiking neural network (SNN) represents another association between RRAM devices and synapses.[59] Applying voltage pulses to an RRAM device can gradually change its conductance. With careful designs, the SET voltage pulses can be used to realize potentiation (excitatory), whereas the RESET voltage pulses enable depressing (inhibitory) functions. Such an RRAM synapse together with an external voltage spike generator can be adopted with SNN learning rules to implement unsupervised learning applications.[60]

Centric to VMM, RRAM arrays can be used to implement the DNN inference functions very efficiently. A DNN model contains many layers of matrices, each of which can be deployed on one or a few RRAM arrays. The similar voltage–current flow mechanism of matrix multiplication has been widely adopted in research studies, whereas weight representation can be realized in an analog or digital manner.[61] In the analog scheme, the synaptic weights directly map as the conductance values. Such an "analog synapse" requires that the device can be programmed to any value in a certain conductance range. In contrast, a digital scheme can accommodate only a limited number of resistance states, denoting different digital levels of weights.[62] Often, a column of RRAM cells is with the same digit-wise significance. The computational results from different columns can be summed by applying the corresponding significances.

Moreover, RRAM arrays have been used for various neural network models with the assist of weight segmentation techniques (Figure 2b).[61,63,64] For example, a fully connected layer can be directly mapped on one RRAM array or partitioned and implemented onto a few smaller arrays. A convolutional layer in convolutional neural networks (CNNs) contains many convolutional kernels (or filters), which need to be unrolled to the VMM format first.[63,65,66] For long short term memory (LSTM) networks, the synaptic weights in an LSTM layer toward input gate, output gate, and forget gate can be deployed on different RRAM arrays.[64] In this design, the input vector and feedback of the output vector are encoded in the voltage format and fed into RRAM arrays in parallel. The intermediate computation results of different gates come from different columns of the RRAM array simultaneously. No matter what types of network models, the execution parallelism can dramatically improve the throughputs.

2.2. RRAM-Based Designs in Training

Most of the RRAM-based DNN accelerator designs target Internet of Things (IoT) and edge computing applications.[67–70] In such energy-efficiency-oriented embedded systems, the necessity of on-chip or online training is arguable. The latest progress in decentralized learning and data privacy demands the realization of local training capability in edge devices. However, the RRAM device uniformity and the large overhead of training circuits remain major challenges.[29,71]

Considering the implementation complexity of training circuitry, the devices featuring linear and symmetric synaptic weight updates are more preferred. Here, the symmetric weight update (**Figure 3**a) denotes the scenario where identical electrical excites result in the same amount change in weight Δw, during both SET and RESET programming. The linear weight update means that the weight change Δw has linear dependency on the electrical excites, regardless of the current resistance state. In reality, however, the weight update of RRAM devices is asymmetric and nonlinear (Figure 3a).[72,77] Some theoretical memristor models reason that nonlinearity is natural and derivable; however, the nonideal linear update can be mitigated by relaxing the strong requirements of training.[72] For example, Chen et al. presented the self-rectifying TaO_x/TiO_2 RRAM, which is only $\approx$3% away from linearity during programming.[72] They modified the online training with voltage spikes and mitigated nonlinearity by fine-tuning the exciting spike widths. As shown in Figure 3a, an approximation of linear and symmetrical

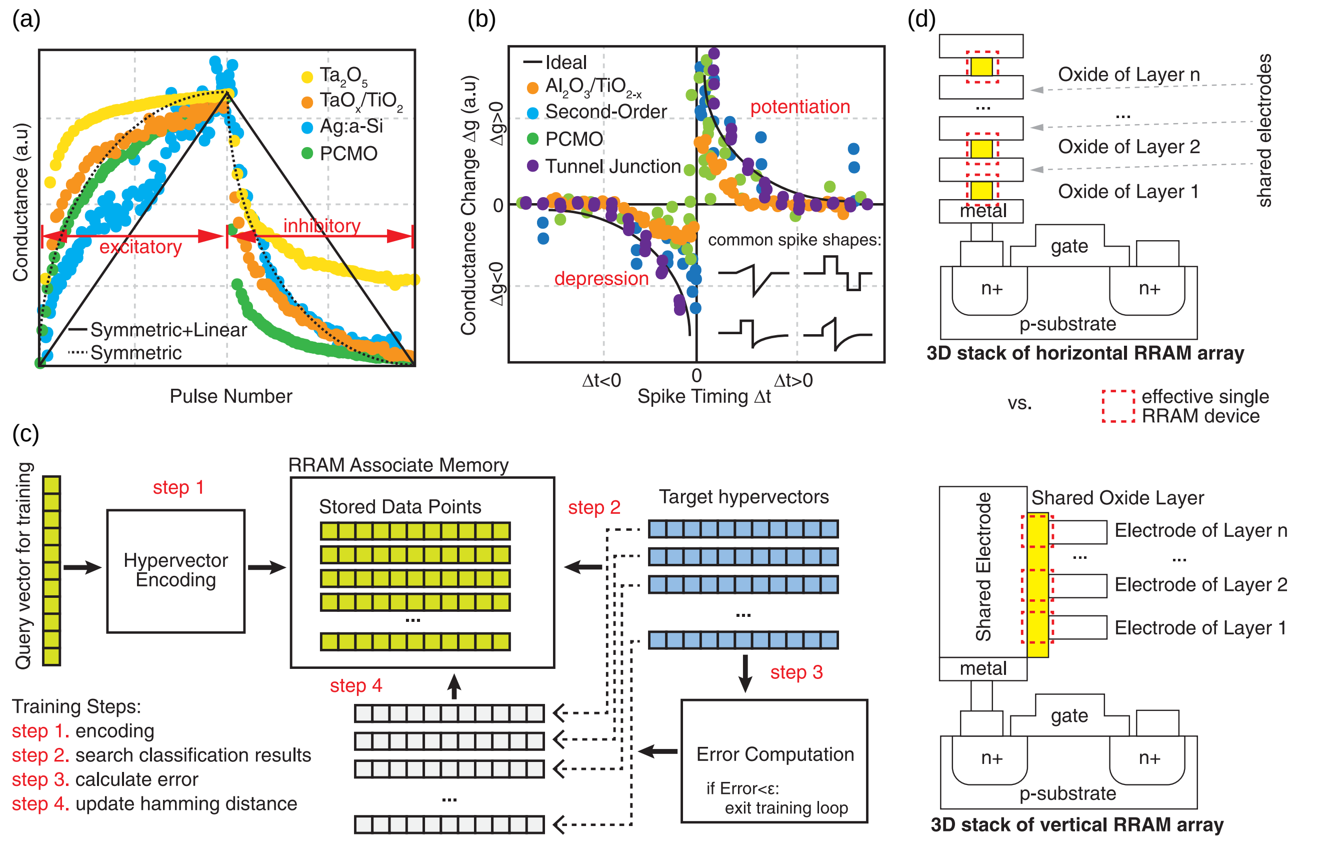


Figure 3. a) Measured weight update by bidirectional identical pulses of Ta_2O_5, TaO_x/TiO_2, Ag:a-Si, and praseodymium calcium manganese oxide (PCMO) memristors.[72,73] All conductance values are normalized to the same scale. b) Measured change of synaptic connections as a function of the relative timing of pre- and postsynaptic spikes using Al_2O_3/TiO_{2-x}, second-order, PCMO, and tunnel junction memristors.[31,55,74,75] All conductance change values are normalized to the same scale. c) Training scheme of HD computing using RRAM associate memory.[76] d) Two schemes of 3D stack RRAM configurations.[77–79]

programming is achieved by modulating the pulse width or voltage potentials. Moreover, Yu et al. demonstrated an online training by fabricating quasi-linear devices and integrating in the 1T1R cell.[80] The weight updates, calculated by a gradient-decent method, are translated into the number of identical programming pulses to RRAM devices. The accuracy of modified National Institute of Standards and Technology (MNIST)[81] dataset reaches 96.5%, which is very close to the one obtained at a software level.

A backpropagation algorithm commonly adopted in neural network algorithms consists of some complex computing operations, such as partial derivatives and outer products.[76,83] In-memory computing modules are the most convenient for inner products but do not support outer products. There exist studies on on-chip learning of RRAM-based design by leveraging its advantages in matrix-multiplication operations.[83] However, it is not feasible to implement the entire backpropagation procedure within in-memory computing modules. The assist of extra circuitry or graphics processing unit (GPU)/CPU cores is necessary.

Backpropagation, however, is not the only choice. With additional controlling and peripheral circuitry, it is feasible to realize other training schemes with RRAM synapse, such as spike-timing-dependent plasticity (STDP).[84] STDP is a biological plausible training method that operates time-related information into weight updates.[21] "Synaptic plasticity," similar to its definition in neuroscience, refers to the synaptic weights that can be strengthened (increased) or weakened (decreased) over time. In STDP configuration, the weight of a target RRAM synapse changes according to the delay of the spikes from the presynaptic neuron (the neuron before the synapse) to the postsynaptic neuron (the neuron after it), as shown in Figure 2a. The "delay" can also be negative, meaning the postsynaptic spike fires before the presynaptic spike. The delay, or more accurately, the time difference, serves as the input to the learning function. The output of the learning function is the weight update of the target synapse. Voltage pulses are generated based on the difference of the presynaptic and postsynaptic spike timing for RRAM synapse updating (Figure 3b). To achieve weight updating according to spike timing, the shape of the spikes applied to the RRAM synapse is critical. Figure 3b summarizes some examples. Most of the current designs produce such spike shapes directly from the testing equipment or by an analog CMOS neuron circuit design. A recent research study shows that RRAM (memristor) devices can also be used to build spike-based neurons, the details of which will be elaborated in Section 3.2.

As an alternate to train RRAM synapse, hyperdimensional (HD) computing eases the computation complexity and enhances training efficiency with interpretability (Figure 3c).[85–87] HD computing encodes the input data into query vectors and compares them with a set of hypervectors trained from various classes. A hypervector is a representative of the characteristic of a specific class. The encoding method reshapes multidimensional input data into a series of scalars and lines them up to form query vectors. Both hypervectors and query vectors are often binary and sparse. As an inference, the associate memory performs the "search" operation of query vectors for its most similar hypervectors.[76] The hypervector with the least hamming distance to the query vector indicates the classification or regression results. There are two advantages when implementing such a comparison process with RRAM arrays. First, the associate memory can be easily built on RRAM arrays.[88] Second, an RRAM array, together with the winner-takes-all (WTA) circuit, can directly deliver the hamming distance calculation. A demonstration of HD computing was presented by Wu et al., which monolithically integrated carbon-nanotube field-effect transistor (CNFET) and RRAM synapses.[87] The training of such a scheme can be simply realized by updating the hypervectors stored in an RRAM-based associate memory and aligning their hamming distances to the demanded results.

2.3. 3D Stacking for Scalability

The capacity of RRAM arrays grows by increasing the array size.[22,67] Furthermore, 3D stacking techniques can expand the array layers vertically with a minimal impact on the die area.[77–79,89] Figure 3d shows the 3D implementation of the RRAM array above the transistors. For example, Adam et al. demonstrated a two-layer stack RRAM array,[78] where the TiO_{2-x} RRAM is fabricated on Si wafer coated with SiO_2. The fabrication process requires a low-temperature profile ($<175\,^{\circ}C$) to prevent from damaging the fabricated horizontal RRAM layer, as shown in Figure 3d. The chip consists of two layers of 10×10 arrays, i.e., 200 RRAM devices in total. The two stackable layers share the middle electrodes. Under the statistical measurement of conductance and programming hysteresis I–V curves, the RRAM devices of the two layers present similar characteristics.[78] The fabrication on the Si wafer indicates its potential for CMOS-compatible monolithic back-end-of-line (BEOL) integration. Though the design adopts a simple RRAM-only cell structure without a selector device, it shows the possibility of integrating multiple RRAM arrays vertically with middle electrodes.

Li et al. first reported a four-layer stack RRAM array monolithically integrated with fin field effect transistor (FinFET).[79] Each layer of the HfO_x RRAM array has a size of 32×32 devices. Different from the aforementioned two-layer 3D RRAM by Adam et al.,[78] this four-layer 3D RRAM array has a vertically integrated RRAM configuration: one TiN/Ti top electrode is shared by the corresponding cells of four adjacent layers, whereas TiN bottom electrodes are separately assigned to each specific layer. As such, a middle electrode is internally connected to the drain/source of the selecting FinFET, forming a one-transistor-four-RRAM (1T4R) high-density cell. Luo et al. showed an example of vertically integrated four-layer 3D RRAM.[90] These devices are naturally with a switching-on threshold to mitigate the sneak path impact due to the lack of cell selecting switches. In this design, the top electrodes determine the input vector dimension; the bottom electrodes and the stack layer number decide the output vector dimension.

The 3D stack RRAM can further advance in-memory computing with bigger storage capacity, more efficient local data processing, and bigger bandwidth and throughput.[38,89,91] The new concept of the 3D synapse also has emerged, using the multiple layer of 3D RRAM arrays to store the synaptic weights and conduct matrix multiplication.[92] It collects the output currents vertically from RRAM devices of different layers.[78] In such a configuration, the number of allowable layers limits the dimension of input vectors. From the perspective of circuit design, the ports and topology of the top/middle/bottom electrodes should be constructed in a similar way as a 2D RRAM with shared electrodes as the VMM outputs. In this way, the analog to VMM is established. The support of parallel or partially parallel operations among different layers is crucial because this is the only way to enhance the bandwidth of the 3D stack RRAM. Meanwhile, careful calibration of the selecting device connection is necessary to avoid sneak paths in programming and sensing.[93]

The Joule heat dispersion is now a concern in developing the 3D stack RRAM. Sun et al. revealed this "thermal crosstalk" in a 3D RRAM array and modeled it quantitatively.[94] The Joule heat generated from one RRAM device may heat its neighbor cells and cause unexpected failures, especially during the RESET operation that requires a relatively large programming current (i.e., more severe ohmic heating) compared with the SET operation. The dense alignment of RRAM devices in a 3D crossbar array deteriorates heat dispersion due to the memory cells that are made very close to each other.

3. RRAM Devices for Neuron Activation

Neuron circuits of a certain neural network layer interface with neighboring neural network layers. Owing to the analog signal processing nature, the CMOS neuron design in analog circuits has lasted for decades.[95–99] In SNNs, integrate-and-fire circuit (IFC) built based upon capacitors and transistors is usually adopted, showing stable performance and robustness. Some recent research works also present the use of emerging nanoscale devices, especially RRAM (or memristor) devices, in emulating biological neuron functions.

Wang et al. proposed to using diffusive memristors featured with stochastic dynamics to construct neuron circuits.[100] A diffusive memristor sandwiches a SiO_xN_y or SiO_x layer that is doped with Ag nanoclusters between two metal electrodes. During SET operations, field-induced Ag mass transportation is formed between the electrodes, and thus, the device gradually changes to an LRS. During RESET operations, the Ag diffusive dynamics dissolves the nanoparticle bridge after a certain characteristic time, and hence, the device is relaxed to an HRS. This conductive process is a combination of both the Ag mass transportation induced by an external electrical field and the conductive filament formation. The special selection of Ag results in a dedicated delay in response to a train of programming spikes. The amount of this delay can be well controlled through carefully selecting the external shunt capacitor. This delay of response is the key feature to emulate the "threshold firing" operation in spiking neurons. More specifically, only inputs larger than a certain neuron threshold lead to output spikes. The "threshold firing" of diffusive memristors behaves in a similar way as ion channel formation in biological neuron cell membranes (**Figure 4**a). Hence, such a resistive SiO_xN_y/SiO_x:Ag memristor, with minimal additional circuit elements, is prominent than CMOS analog neuron circuits to mimic the IFC function. Compared with the analog IFC circuit, the diffusive memristor neuron circuit occupies a much smaller area and consumes much less power.[100,103] Together with RRAM synapses (Section 2.1), it is possible to realize a "fully memristive network," in which both synapses and neurons are RRAM devices.

The Hodgkin–Huxley neuron model is used to explain the dynamics of biological axons via electrical elements.[104] Biological neurons process signals by mediating the sodium and potassium ion channels. The procedure is abstracted as the conductance varying over time in the Hodgkin–Huxley neuron model.[104] This model can be emulated physically by co-operating the Mott memristor with capacitors, which forms a set of analog neuron circuits named "neuristors."[40] Mott memristor is a type of RRAM nanoscale device whose hysteresis (memory) loop is formed based on Mott transition, a reversible insulation-metal phase transition.[40,101] This transition, often activated by certain thermal conditions, also causes the negative differential resistance (NDR) phenomenon. That is, the induced current decreases as the applied voltage increases. This uncommon nonlinearity has been leveraged in developing a relaxation oscillation circuit. When a Mott memristor in the NDR region is connected to a resistance–capacitor (RC) charging circuit (Figure 4b), the Mott memristor could force charges to flow toward (away from) the capacitor even as the capacitor discharges (charging). The reciprocating trend of charge flow between the Mott memristor and capacitor results in a sawtooth-shaped current oscillation, even the capacitor is supposed to be charged only under the excites of a direct current (DC) voltage source. The Mott memristor-based neuristor circuit exploits the oscillating dynamics to generate output spikes. It consists of two sets of Mott memristor-RC circuits coupled with each other (Figure 4c).[40] The design demonstrates similar neuron behavior described in the Hodgkin–Huxley neuron model and presents the "threshold firing" function. Compared with diffusive memristors that process spikes only, a Mott memristor-RC circuit can spawn spikes under DC excitation. In other words, Mott memristor-based neuristors integrate both "threshold firing" and spike generation capability, which advances the diffusive memristor at the cost of more complex circuit connections. Experiments show that the NbO_2 Mott memristor-based neuristor achieves rapid

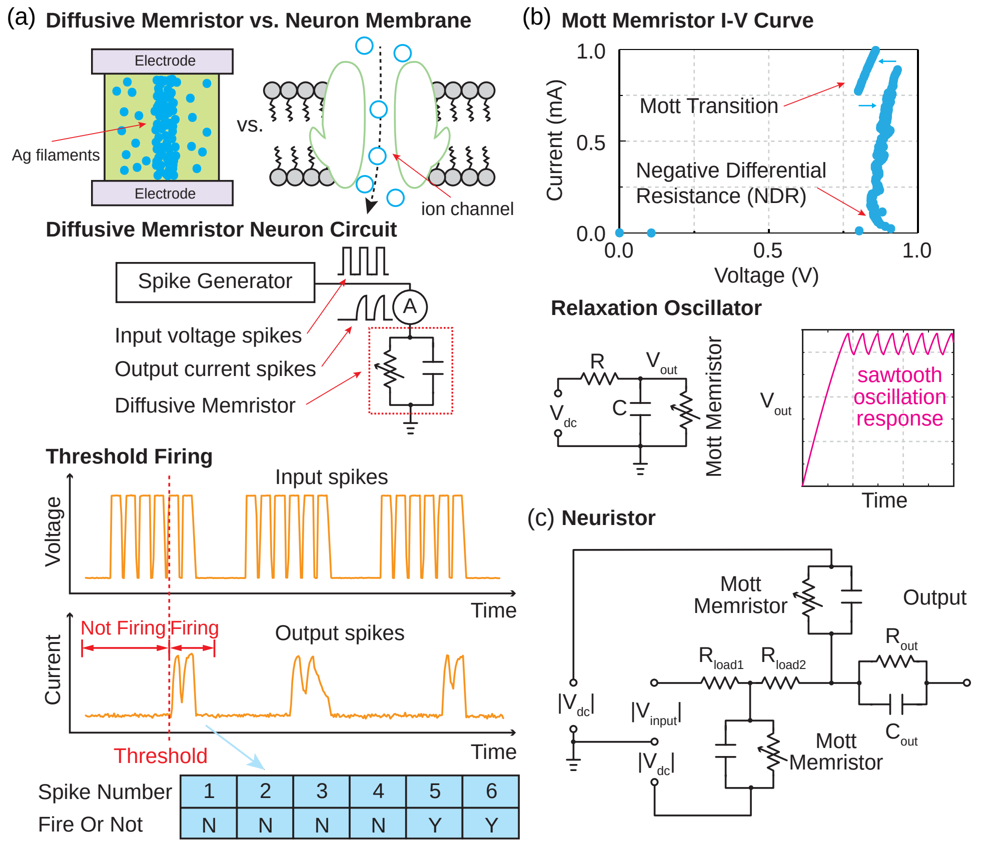


Figure 4. a) The SiO_xN_y/SiO_x:Ag diffusive memristor is similar to the neuron cell membrane in the conductive channel formation.[100] A diffusive memristor-based neuron circuit demonstrates the "threshold firing" function.[100] b) Mott memristor characteristics showing Mott transition and NDR.[101,102] The step response of the Mott memristor-RC circuit is sawtooth-shaped spikes. c) The neuristor circuit contains two groups of Mott memristor-RC circuits.[40]

spike generation ($\leq$1 ns), very low switching energy (<100 fJ), and a much more compact design ($110 \times 110\,nm^2$).[40]

To reduce the routing complexity in the neuristor, Yan et al. proposed to integrate both "threshold firing" and spike generation functions within a group of Mott memristor-RC circuits.[102] With an appropriate post-amplifier to fit into the voltage range of CMOS logic, a single Mott memristor-RC oscillation circuit can replace the analog IFC in conventional CMOS technology (Figure 4c). Additionally, the Mott memristor shows the quasi-chaotic behavior, i.e., intrinsic pseudo-randomness. Adding the random noise with limited amplitude to the outputs helps jump out of the local minima in the backpropagation process and thus improve training speed.[101,102] The experiment showed that online training with the Mott memristor neuron circuit is 1.8× faster on average than the design with the analog CMOS IFC. For a fully connected layer benchmark, the RRAM in-memory computing macro saves 27% area and reduces 36% power consumption.[102]

In addition to developing high-density and a large volume of synaptic connections, the neuron design is another key to the efficient implementation of in-memory computing accelerator design. In recent years, RRAM-based neurons have gained substantial attention as RRAM devices feature high density, low power consumption, and easiness to emulate complex neuron dynamics. The intrinsic connection between biological nervous systems and memristors is proved and explained in theory.[105,106] Meanwhile, new types of RRAM neuristors are brought to this emerging field.

4. Large-Scale System Integration

Neural network model sizes surge as deep learning methodologies prevail in solving the recognition and regression problem. How to implement large-scale in-memory computing systems becomes very important. For RRAM-based large-scale systems, the peripheral circuitry together with the RRAM arrays often dominate the overall power, chip area, and energy consumption. Thus, new timing control and data conversion circuitries are expected. Furthermore, more complicated topologies of neural networks require considerable data management and scheduling for better exploiting in-memory computing systems. In this section, we present both in-memory computing macro designs with different data conversion interface circuits and state-of-the-art microarchitectures of RRAM-based in-memory computing.

4.1. RRAM-Based In-Memory Computing Macro Design

4.1.1. Analog/Digital Converter-Based Design

An in-memory computing macro is the basic processing core of VMM operations. RRAM-based in-memory computing indeed operates in an analog format and requires data conversion for such a macro to interface with its surrounding digital systems. Mature digital/analog converter (DAC) and analog/digital converter (ADC) designs become the first choice. Hu et al. presented a pure analog dot-product engine (DPE) using a 128 × 64 RRAM array.[61] The digital inputs vectors are converted into queues of analog voltages, and DPE performs computation. The output current is converted to voltage by a transimpedance amplifier and subsequently translated into a digital output vector by ADC. The DPE works at a frequency of 10 MHz limited by the multiple conversion paths and high parasitics at the 2 μm transistor technology. The Ta/HfO_2 RRAM is with 6 bit precision, and a single-layer perceptron using such a DPE for performance MNIST dataset recognition yields an accuracy of 89.9%.

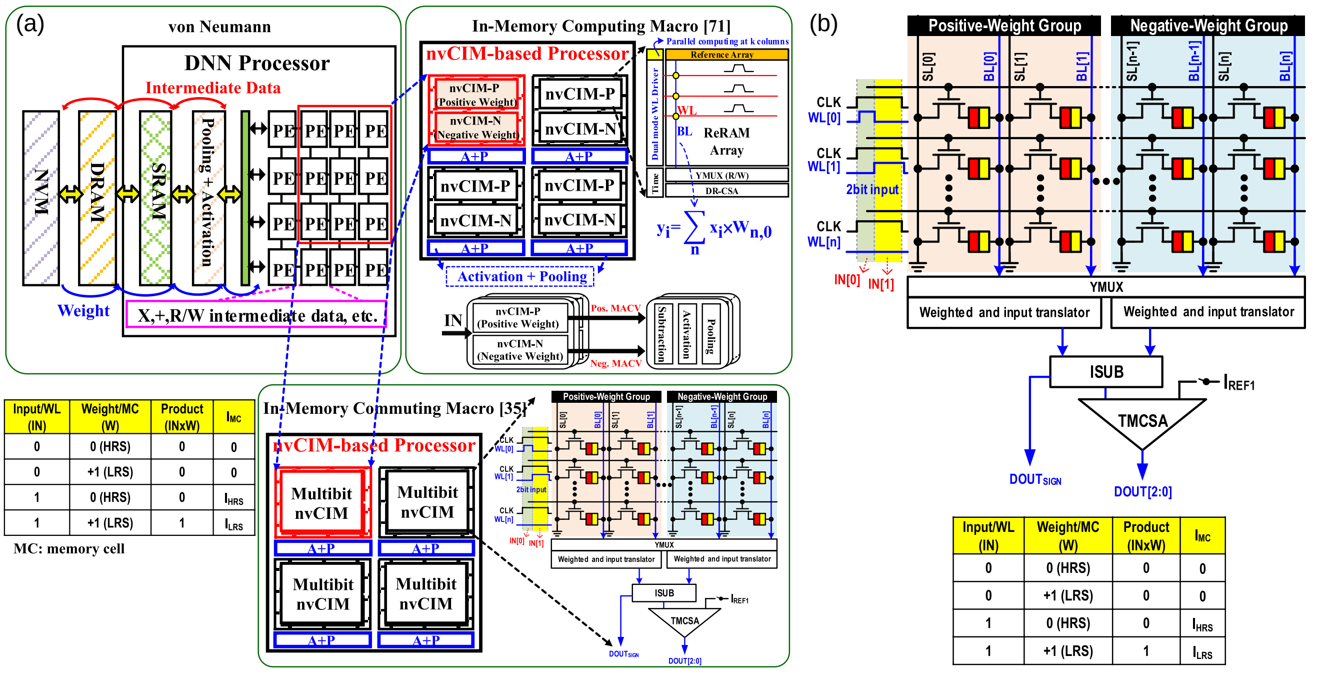
4.1.2. Level Sense-Amplifier-Based Design

The pure analog approach of DPE using the RRAM array requires substantial efforts to fine-tune the analog RRAM cell resistance.[67] In contrast, digital approaches can simplify the data conversion inside the in-memory computing macro. A closer look at the RRAM-based in-memory computing macro reveals the three parts of digitalization based on vector multiplication: the two operands $\boldsymbol{G}$ and $\boldsymbol{V}$ and the computational results $\boldsymbol{I}$. To digitalize $\boldsymbol{V}$, a multibit digital input is applied to the RRAM array bit by bit. Each bit is with an identical voltage. The results are subsequently weighted with significance and summed. In this way, DAC at the input terminal is removed at the cost of increased computing cycles. This method is often used to isolate the input drivability and input data. Otherwise, DAC with a special drivability requirement consumes much more area.

The quantization of $\boldsymbol{G}$ determines mapping from a floating-point weight in neural networks to RRAM conductance with a limited number of levels. The state-of-the-art monolithic integration of RRAM in an order of kilobits and more onto the CMOS logic platform relies on binary RRAM devices, which demands the adaption of floating-point synaptic weights to binary or ternary ones. Wang et al. presented a few schemes, including distribution-aware quantization, quantization regularization, and bias tuning, to adapt synaptic weights during training to fit into RRAM in-memory computing macro.[107]

The lowered requirement of up to 4 bit output precision (a.k.a. activation precision) primarily simplifies the ADC design. Instead of conventional ADC architectures (such as pipelined ADC and successive-approximation ADC), low-precision ADCs are feasible from multiple binary sensing amplifiers with different reference thresholds to address.[108] Mochida et al. presented a binary-input binary-output RRAM-based in-memory computing macro.[67] The RRAM array is composed of 1T1R cells. Assisted with the read-verify programming scheme, an RRAM device can be programmed to any value between its HRS and LRS.[67] The output sensing amplifier is binary. For neural networks requiring multibit activation precision, neuron computation is then realized by accumulating the 1 bit MAC results followed by additional digital nonlinear activation function circuits. Such a simplified design scheme without an ADC module reduces the area and power consumption with increased latency/operation overhead.

Chen et al. demonstrated a binary-input 3 bit output RRAM in-memory computing macro based on the single-level cell (SLC) RRAM.[109] **Figure 5**a shows its architecture. The binary input signal is determined by turning on the word line, and the weights are stored in the memory array. There are two states in RRAM cells, HRS and LRS, respectively, representing the weights of +1 (LRS) and 0 (HRS). There are two RRAM in-memory computing



Input/WL (IN)	Weight/MC (W)	Product (INxW)	I_{MC}
0	0 (HRS)	0	0
0	+1 (LRS)	0	0
1	0 (HRS)	0	I_{HRS}
1	+1 (LRS)	1	I_{LRS}

MC: memory cell

Input/WL (IN)	Weight/MC (W)	Product (INxW)	I_{MC}
0	0 (HRS)	0	0
0	+1 (LRS)	0	0
1	0 (HRS)	0	I_{HRS}
1	+1 (LRS)	1	I_{LRS}

Figure 5. a) Architecture of binary-in ternary-weight RRAM in-memory computing macro. Reproduced with permission.[109] Copyright 2018, IEEE. b) Architecture of serial-input nonweighted product RRAM in-memory computing macro. Reproduced with permission.[68] Copyright 2019, IEEE.

macros that, respectively, provide the positive and negative weights to reach the ternary weight. This RRAM in-memory computing macro yields a latency of 14.8 and 15.6 ns to compute a convolutional layer and a fully connected layer, respectively. The precision of the sensing circuit is 3 bits. Moreover, it uses an input-aware reference current generation to increase the read margin. A small-offset multilevel current sense amplifier improves the sensing yield. Xue et al. further optimized the sensing circuit to a 4 bit precision with 14.6 ns latency.[68] Figure 5b shows the structure of the RRAM in-memory computing macro using a 1T1R SLC cell array, including a serial input unweighted product array structure, a read path current reduction module, and a multilevel current mode sense amplifier. This work allocates positive and negative weights in different columns in the same array, which is realized by a current subtractor. Besides, multiple binary RRAM devices together represent a multibit synaptic weight. For an $N \times N$ CNN kernel, N^2 weights are stored in N^2 consecutive rows.

In-memory computing appears to be a promising approach, using large memory internal bandwidth and enabling parallel data processing in the local memory. The in-memory computing structure also has several advantages over the conventional approach. First, in-memory computing reduces the amount of data that must be transferred between the CPU and memory. Second, it reduces the amount of intermediate data, which decreases memory capacity requirements, reduces energy consumption, lowers latency, and improves overall performance. To accommodate the higher precision requirement of heavy DNN applications, RRAM in-memory computing macro has to support multibit inputs and weights to maximize the accuracy of the MAC output.

4.1.3. Spike-Based Design

Although there are many algorithm-layer studies to develop a purely binary neural network (binary input, binary output, and binary weights),[110,111] complicated datasets and applications, e.g., ImageNet,[112] still need a certain level of data precision (e.g., 8 bit) to satisfy the accuracy requirement.[113] Considering the limited sensing margin in voltage representation of data,[68,109] Yan et al. proposed using spikes for data representation and demonstrated a compact RRAM-based nonvolatile in-memory computing processing engine (PE).[114] The 1T1R array is 64 kb (256×256) with RRAM devices in binary states. The PE has the duality function of memory and computation. In the memory mode, the read/write logics, drivers, and amplifiers realize data programming and sensing. In the computing mode, the RRAM array performs matrix multiplication, and the in situ nonlinear activation (ISNA) circuit converts the output currents to spikes. In this in-memory computing PE design, ISNA executes the activation function computation on the fly, obviating the additional circuits to calculate activation function and reducing the design overhead.[115] The spike-based ISNA takes a different approach to enhance the energy efficiency by lowering the power consumption at the cost of ≈200 ns latency. Instead of using multiple sensing amplifiers with different thresholds,[68,88] the IFC-like ISNA circuit performs data conversion by continuous charging and discharging of a capacitor. Such a biological-inspired spike generation needs only approximately ten transistors with a capacitor. Because of the small footprint of ISNA, more spike-based ISNA sensing circuits can be included, leading to a higher execution parallelism and bigger throughput. This RRAM in-memory computing PE reaches the highest energy efficiency of 16 trillion operations per second per watt (TOPS/W) as well as provides the flexibility of configuring the activation precision between 1 bit and 8 bits.

4.2. RRAM-Based In-Memory Computing Microarchitecture

According to von Neumann, a computer can be divided into basic arithmetic operations and logic flow.[1] In-memory computing macro designs provide the function of the basic arithmetic operation. Microarchitecture studies focus on effective control to utilize in-memory computing macros.

ISAAC is a crossbar-based accelerator tailored for CNN benchmarks (**Figure 6**).[116] It is organized in a hierarchy of chips/tiles/in situ multiply accumulators (IMA)/arrays. The dedicated on-chip network bridges the tiles within the chip for data transmission. Within each tile, embedded dynamic random access memory (eDRAM) buffers are used for result aggregations, IMAs are composed of a group of RRAM arrays together with data conversion interface and conduct VMM, and the output registers store the aggregated results. Additional digital components perform the pooling and activation operations in neural networks.[116] ISAAC exploits the characteristics of networks and proposes a pipeline design. The pipeline is applied within IMA and tiles and enables the overlap of data accesses and computations. ISAAC also equips with the data encode and allocation scheme to lower the overhead induced by high-precision DAC/ADCs.

In the same year, another neural network accelerator, PRIME, was published by the research group from University of California, Santa Barbara (UCSB).[117] Different from the hierarchical structure introduced by ISAAC, PRIME was built upon the traditional main memory architecture, so the overhead of design modification is minimal. In the design, the peripheral circuits of a portion of RRAM arrays are enhanced to support the computing functions. These arrays can alter between memory and computation modes in a time-multiplexed manner. A large amount of data can reside in RRAM arrays instead of in external memory, which reduces the overhead of memory and data access. Furthermore, PRIME provides a set of software and hardware interfaces, such that the RRAM arrays are configured into memory or computing units according to the application demand.

PipeLayer enhances the execution parallelism across two levels, i.e., intra-layer parallelism and inter-layer parallelism.[118] Furthermore, it removes the high-cost ADC/DAC components and replaces with spiking-based read/output circuits. More importantly, PipeLayer implements the customized processes for training, i.e., error backward and weight update. Integrating the aforementioned designs, PipeLayer significantly improved the energy efficiency, the computing throughput, as well as the area efficiency. Based on the observation that the existing RRAM accelerators overlooked the data reuse opportunity underlying the network layer, Qiao et al. proposed a universal accelerator, AtomLayer, which integrates a unique filter mapping scheme and a dataflow design to maximize the utilization of input data and execution throughput.[119] The performance and power efficiency of AtomLayer exceed the previous works in both training and inference.

Recently, a few RRAM-based in-memory computing specialized for diverse network networks have emerged. LerGAN and ZARA are tailored for accelerating the unsupervised machine learning applications, generative adversarial network (GAN).[120,121] The challenge of training GAN comes from two aspects: 1) the complex data dependency between the discriminator network and the generator network and 2) the untraditional computing patterns within the layers of the generator network. To address these problems, LerGAN derives the chances to improve the computing efficiency by skipping the ineffective computations in a generator network.[121] Meanwhile, a 3D-based layer connection is developed to optimize the efficiency of data transmission among the layers of the discriminator and generator. Regarding the same problems, ZARA emphasizes the computing efficiency optimization.[120] It first decomposes the convolution in the generator into several submatrix multiplications and then balances their computation latency through weight mapping and execution scheduling designs. By eliminating the zero-related ineffective computation, ZARA achieves almost 2.1× performance over the previous RRAM-based in-memory computing accelerators.

Furthermore, to enable the general purpose application of the RRAM-based microarchitecture, Ankit et al. proposed programmable ultra-efficient memristor-based accelerator (PUMA) architecture with an instruction set architecture (ISA) and compiler for a wide variety of machine learning workloads.[122] The PUMA ISA accommodates the hardware design configuration and provides an interface for the up-level compiler. Computing instructions include VMM, vector arithmetic, and

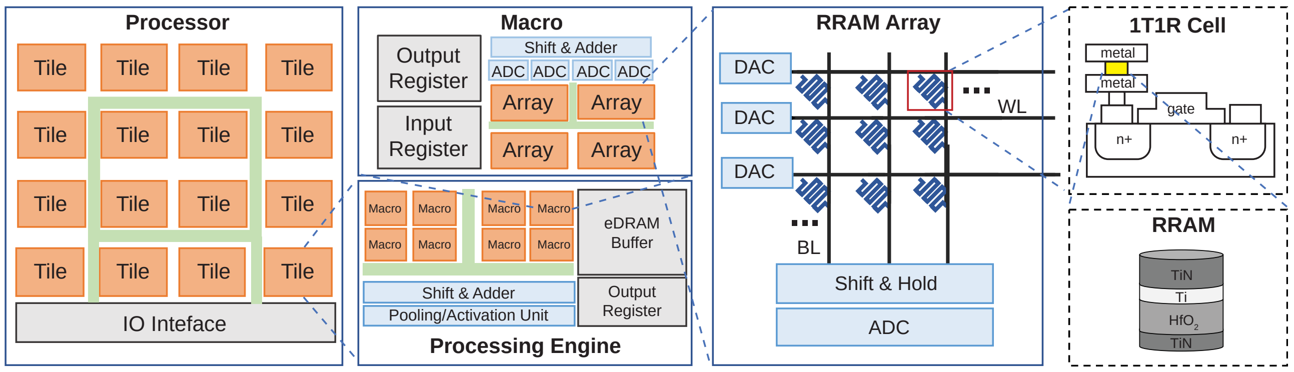


Figure 6. Hierarchy of RRAM in-memory computing microarchitecture: from top-level to bottom-level is processor, PE, macro, RRAM array, 1T1R cell, and RRAM. The data conversion shown implemented with DAC/ADC. Reproduced with permission.[51] Copyright 2019, ACM, Inc.

scalar arithmetic. Each of the three-level memory hierarchies has its own controlling instructions, which, respectively, are set/copy, load/store, and send/receive, from low to high. The argument lists of all aforementioned operations are directly encoded in the instructions. The gap between the machine code and high-level machine learning (ML) model descriptions is fulfilled by the PUMA compiler. It directly compiles ML models written in popular frameworks (Caffe2, PyTorch, etc.) to executable PUMA instructions.[123–125] Note that PUMA is a spatial architecture, for which each core has its own sequence of instructions. The first stage of compiling is, therefore, to derive the code for each core. A machine learning model is described as a computation graph where a node represents an operation and an edge represents a communication. A heuristic-based graph partition is performed to assign graph nodes to PUMA cores and replace edges with load/store/send/receive operations. Next, the compiler schedules the instructions for each PUMA core. Dataflow analysis techniques are applied to reduce register pressure, capture instruction-level parallelism, and avoid deadlocks. Finally, register allocation is performed to fit the actual hardware.

5. Reliability of RRAM-Based In-Memory Computing

Reliability becomes as a major concern for large-scale integration. For the RRAM-based in-memory computing systems, the reliability issues are induced by not only device fabrication but also the computing process. First, process variations cause the devices across a single chip (within the same array or on different arrays) to behave differently in terms of the conductance ranges, device programmability, retention, and endurance. The heterogeneity of RRAM array fabrication potentially increases faults and reduces the yield. The variations across different chips are even more severe. Moreover, nonoptimized operational behaviors, such as repeatedly rewriting a small portion of devices, could result in overall system performance deterioration.[84] These nonideal properties are summarized in **Table 1**. Some nonideal properties can be exploited for special purposes, such as developing physically unclonable functions (PUF) with the statistical variance of RRAM devices.[129] In general, the robustness concerns of RRAM-based in-memory computing obscure the accuracy of both storage and computation. Comparing RRAM in-memory computing systems with conventional storage-purpose RRAM designs, the inaccuracy of devices not only affects data storage, but also the computing accuracy of analog matrix multiplication. Due to the highly parallel operations in an RRAM in-memory computing macro, conventional techniques, like error-correcting code (ECC), are not sufficient to tolerate faults without forfeiting the throughput bonus brought by parallelism.

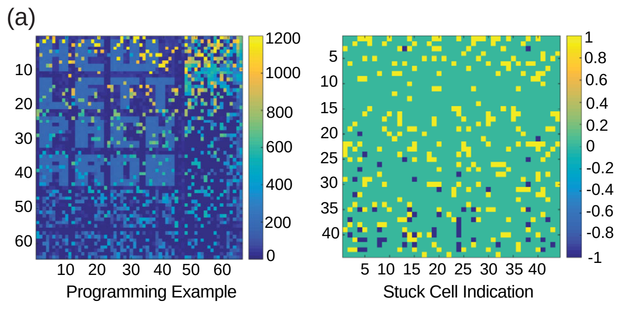
Table 1. Toy model for RRAM nonideal properties.

Nonideal behavior	Analog RRAM	Binary RRAM
Endurance[22,80]	500 k cycles 95.5% maintains resistance	$>10^6$ cycles
Variability[a)][126]	≈0.03	≈0.04 @ LRS, ≈0.4 @ HRS
Yield[61,80]	89.9%	>99%
Bit error rate before ECC[47]	N/A	$<10^{-5}$
Thermal-activated fluctuation variability[a)][94]	≈0.03	≈0.03
Read disturbance	Refer to Yan et al.[127]	Refer to Ho et al.[128]

a)The variability is defined as the ratio of the standard deviation over the mean of the measured resistance.

When addressing the reliability concerns, fault models are used to understand the persistent and nonpersistent errors of RRAM arrays. For example, Ambrogio et al. used $1/f$ noise and telegraph noise models to describe the low-frequency noise in binary RRAM devices.[130] Huang et al. presented an analytic model on RRAM retention properties.[131] Chen et al. described the endurance of RRAM devices toward repeated writing.[132] There is no universal model that can cover all types of RRAM devices, due to a large variety of materials and structures. Nevertheless, the investigations on fault models for different types of RRAM devices unveil some common characteristics,[133] which are helpful to circuit and system designers to understand and explore reliability enhancement technologies for RRAM in-memory computing systems.

Specifically, one of the prominent problems is the low yield of RRAM devices. As shown in **Figure 7**a, many devices in an RRAM array always are in LRS ("stuck-on") or in HRS ("stuck-off"). These devices cannot be mapped to any arbitrary values when deploying an algorithm on an RRAM array. This issue is especially vital to large-scale RRAM arrays, because a



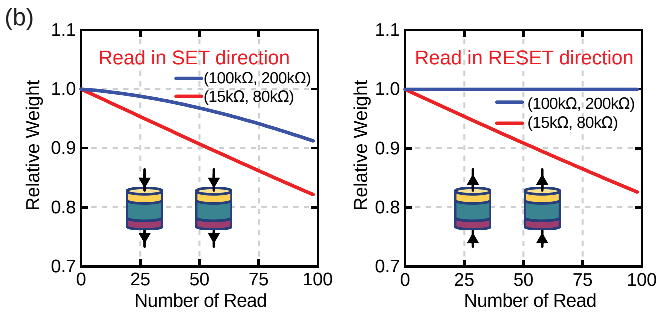


Figure 7. a) Yield problem. Left: conductance heat map in a programming example (64 × 64 arrays). Right: measured stuck-on (+1) and stuck-off (−1). Reproduced with permission.[134] Copyright 2017, IEEE. b) Read disturbance causes weight drift (simulated). A synaptic weight is represented by the conduct difference of two RRAM devices. The shown two cases cause combined weight to decrease. Reproduced with permission.[127] Copyright 2017, IEEE.

small proportion of devices in a large RRAM array still covers considerable MAC operations. To mitigate the low yield issue, Liu et al. introduced the concept of "weight significance," which evaluates how severe the impact of the unexpected deviation of these weights is on the final computational accuracy.[134] By assigning less weight significance to stuck-on or stuck-off cells, the computational accuracy can be largely recovered from neural network deployment on RRAM arrays with lower yield. Xia et al. presented an online training scheme, which can detect and remap the erroneous cells.[135] The detection uses an adaptive threshold voltage to locate the stubborn stuck cells. The remapping technique exchanges the neuron positions to bypass those dead cells recognized from detection.

Faults also arise in the process of data movement and computation. The read-verify programming scheme effectively lowers the bit error rate to the order of 10^{-5} for binary RRAM and suppresses the programming conductance error under 2.95% for analog RRAM.[47,67] However, RRAM sensing is far more frequent than programming (Figure 7b). Long time use of RRAM in-memory computing macro repeatedly applies voltages to cells, which likely causes read disturbance, which denotes the unexpected weight drift from the original well-trained values. Considering that RRAM is programmed with bipolar voltages/currents, adaptively alternating the sensing direction can effectively mitigate the read disturbance.[127] Such a task can be completed by a feedback controller.[127] With the sensing direction determined by mimicking the training backpropagation feedback, the weight stability improves averagely 14.9×.[127]

6. Conclusion and Remarks

In this paper, we summarize the state-of-the-art progress in developing RRAM-based in-memory computing systems, from device to system layers. The major focus at the device level is to realize electric synapses and neurons with a single RRAM device and/or through simple circuitry that leverages novel device structures. Further enhancing the density and scalability, e.g., by taking advantage of 3D integration, will continue to be an important trend. At the circuit and architecture levels, the RRAM-based in-memory computing that naturally integrates data storage and processing operations is widely investigated. Substantial research efforts focus on improving energy efficiency, reducing the design cost, meanwhile satisfying the system requirements, e.g., operation speed, throughput, and accuracy. New ISA and compilers emerge as the new topic to generalize in-memory computing modules to various applications. Finally, the robustness issue emerges and could be even aggravated as the dimension of RRAM devices scales down and the density tops out.

The adoption of emerging RRAM technology demonstrates great potential in realizing more efficient computing systems, particularly for cognitive applications. To enable large-scale integrated systems for real-world applications, however, there are still many key challenges to be solved, such as the imbalance between the limited device number and the increasing size of neural network models,[136] the difficulty in realizing online training schemes, the automation flow to transfer an algorithm to the given hardware platform, as well as the device robustness and system reliability issues. It is impossible to overcome them

Table 2. Cross-layer design considerations of RRAM in-memory computing.

Work	Layer of major contribution	Data conversion	Hierarchical level[a)]	Computation model	Targeting application/dataset
Balatti et al.[55]	Device	Sense amplifier	Cell/macro	Logic	Logic gate
Pedro et al.[60]	Device	IFC	Cell	SNN	Image clustering
Hu et al.[61]	Device	DAC/ADC	Cell/macro	DNN	DCT,[b)] MNIST[81]
Wang et al.[100]	Device	Memristive neuron	Cell	SNN	Pattern classification
Li et al.[64]	Device	DAC/ADC	Macro	LSTM	Gait recognition
Kumar et al.[101]	Device	Memristive neuron	Cell	SNN	Backpropagation
Hu et al.[58]	Circuit/microarchitecture	DAC/ADC	PE	BSB	Digit recognition
Wu et al.[137]	Device/circuit	Sense amplifier	PE	HD computing	Language recognition
Imani et al.[76]	Circuit	Sense amplifier	PE/microarchitecture	HD computing	Language recognition
Chen et al.[109]	Circuit	Sense amplifier	Macro	DNN, CNN	MNIST
Xue et al.[68]	Circuit	Sense amplifier	Macro/PE	DNN, CNN	CIFAR-10[138]
Yan et al.[114]	Circuit	ISNA	PE	DNN, CNN	MNIST, CIFAR-10
Shafiee et al.[116]	Microarchitecture	DAC/ADC	Processor/macro	DNN, CNN	ImageNet[139]
Song et al.[118]	Microarchitecture	IFC	Processor	DNN, CNN	Image recognition, NN[c)] training
Chen et al.[120]	Microarchitecture	DAC/ADC	Processor	GAN	Image synthesis
Mao et al.[121]	Microarchitecture	DAC/ADC	Processor	GAN	Image synthesis
Ankit et al.[122]	Microarchitecture	DAC/ADC	Processor/compiler	DNN/LSTM/CNN	Object detection, neural machine translation, language modeling, image recognition

[a)]Hierarchical level refers to Figure 6; [b)]DCT: Discrete cosine transform. [c)]NN: Neural network.

solely at a device level, while mitigation from circuits, systems, and even algorithms might be the only solution.

Nowadays, to develop a highly efficient computing system, all the hierarchical layers, including device processing, the circuit components, the microarchitecture, as well as application algorithms, are heavily correlated. The same philosophy is well represented by the research on RRAM-based in-memory computing, as shown by examples at different layers in **Table 2**. As an interdisciplinary area, researchers with different knowledge backgrounds present different understandings on how to develop in-memory computing. For instance, the expansion of computing units leads to the revision from the original classical von Neumann architecture for higher efficiency and higher performance, whereas the unprecedent biological-plausible computing schemes emerge to respond to the ever-increasing AI computing demands. RRAM in-memory computing has attracted the attention from industries and will certainly brace further development of computing power.

Acknowledgements

This work was supported by Air Force Research Laboratory (AFRL) FA8750-18-2-0121 and National Science Foundation (NSF) CSR-1717885.

Conflict of Interest

The authors declare no conflict of interest.

Keywords

accelerators, in-memory computing, neural networks, process-in-memory, resistive memory

Received: June 30, 2019
Revised: July 25, 2019
Published online: September 20, 2019

[1] J. Von Neumann, *The Computer and the Brain*, Yale University Press, New Haven, **2012**.
[2] J. L. Hennessy, D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Elsevier, Burlington, MA **2011**.
[3] D. Kuzum, R. G. D. Jeyasingh, B. Lee, H.-S. P. Wong, *Nano Lett.* **2011**, *12*, 2179.
[4] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, D. Brooks, ACM/IEEE Int. Symp. Comput. Archit. IEEE, Piscataway, NJ, **2016**, pp. 267–278.
[5] T. Potok, C. Schuman, R. Patton, T. Hylton, H. Li, R. Pino, Neuromorphic Computing, Architectures, Models, and Applications. A Beyond-CMOS Approach to Future Computing, The Department of Energy (DOE) Office of Scientific and Technical Information (OSTI), Oak Ridge, TN, June 29–July 1, 2016.
[6] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, M. Zhang, *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 1920.
[7] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, D. Burger, Int. Symp. Comput. Archit. ISCA, Los Angeles, CA, **2018**, pp. 1–14.
[8] J. Zhang, Z. Wang, N. Verma, *IEEE J. Solid-State Circuits* **2017**, *52*, 915.
[9] W.-S. Khwa, J.-J. Chen, J.-F. Li, X. Si, E.-Y. Yang, X. Sun, R. Liu, P.-Y. Chen, Q. Li, S. Yu, M. F. Chang, IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco, CA, **2018**, pp. 496–498.
[10] M.-F. Chang, C.-F. Chen, T.-H. Chang, C.-C. Shuai, Y.-Y. Wang, H. Yamauchi, IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco **2015**, pp. 1–3.
[11] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, D. B. Strukov, *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 4782.
[12] D. Fick, A Peek Into Software Engineering at Mythic, https://medium.com/mythic-ai/a-peek-into-software-engineering-at-mythic-1b0ca5522868 (accessed: June 2019).
[13] S. Jain, A. Ranjan, K. Roy, A. Raghunathan, *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *26*, 470.
[14] D. Fan, S. Angizi, Z. He, IEEE Comput. Soc. Annu. Symp. VLSI, IEEE, Kyoto **2017**, pp. 683–688.
[15] Y. Wang, H. Yu, L. Ni, G.-B. Huang, M. Yan, C. Weng, W. Yang, J. Zhao, *IEEE Trans. Nanotechnol.* **2015**, *14*, 998.
[16] R. Venkatesan, M. Sharad, K. Roy, A. Raghunathan, Des. Autom. Test Eur. Conf. Exhib. **2013**, IEEE, Grenoble, pp. 1825–1830.
[17] S. Kim, N. Sosa, M. BrightSky, D. Mori, W. Kim, Y. Zhu, K. Suu, C. Lam, IEEE Int. Electron Devices Meet, IEEE, Washington, DC **2013**, pp. 30–37.
[18] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, E. Eleftheriou, *Nat. Electron.* **2018**, *1*, 246.
[19] D. Ielmini, H.-S. P. Wong, *Nat. Electron.* **2018**, *1*, 333.
[20] S.-S. Sheu, P.-C. Chiang, W.-P. Lin, H.-Y. Lee, P.-S. Chen, Y.-S. Chen, T.-Y. Wu, F. T. Chen, K.-L. Su, M.-J. Kao, K. H. Cheng, IEEE Symp. VLSI Circuits, IEEE, Kyoto **2009**, pp. 82–83.
[21] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, W. Lu, *Nano Lett.* **2010**, *10*, 1297.
[22] W.-H. Chen, W.-J. Lin, L.-Y. Lai, S. Li, C.-H. Hsu, H.-T. Lin, H.-Y. Lee, J.-W. Su, Y. Xie, S.-S. Sheu, M. F. Chang, IEEE Int. Electron Devices Meet, IEEE, San Francisco, CA, **2017**, pp. 22–28.
[23] A. Sengupta, P. Panda, P. Wijesinghe, Y. Kim, K. Roy, *Sci. Rep.* **2016**, *6*, 30039.
[24] X. Zhang, W. Cai, X. Zhang, Z. Wang, Z. Li, Y. Zhang, K. Cao, N. Lei, W. Kang, Y. Zhang, K. Cao, N. Lei, W. Kang, Y. Zhang, H. Yu, *ACS Appl. Mater. Interfaces* **2018**, *10*, 16887.
[25] Y. Huang, W. Kang, X. Zhang, Y. Zhou, W. Zhao, *Nanotechnology* **2017**, *28*, 08LT02.
[26] K. Cao, W. Cai, Y. Liu, H. Li, J. Wei, H. Cui, X. He, J. Li, C. Zhao, W. Zhao, *Nanoscale* **2018**, *10*, 21225.
[27] W. Kang, Y. Huang, C. Zheng, W. Lv, N. Lei, Y. Zhang, X. Zhang, Y. Zhou, W. Zhao, *Sci. Rep.* **2016**, *6*, 23164.
[28] J. Y. Seok, S. J. Song, J. H. Yoon, K. J. Yoon, T. H. Park, D. E. Kwon, H. Lim, G. H. Kim, D. S. Jeong, C. S. Hwang, *Adv. Funct. Mater.* **2014**, *24*, 5316.
[29] S. Yu, *Proc. IEEE*, **2018**, *106*, 260.
[30] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, G. W. Burr, *J. Phys. D. Appl. Phys.* **2018**, *51*, 283001.
[31] Q. Xia, J. J. Yang, *Nat. Mater.* **2019**, *18*, 309.
[32] D. S. Jeong, K. M. Kim, S. Kim, B. J. Choi, C. S. Hwang, *Adv. Electron. Mater.* **2016**, *2*, 1600090.
[33] D. S. Jeong, C. S. Hwang, *Adv. Mater.* **2018**, *30*, 1704729.
[34] D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, *Nature* **2008**, *453*, 80.
[35] L. Goux, Y.-Y. Chen, L. Pantisano, X.-P. Wang, G. Groeseneken, M. Jurczak, D. J. Wouters, *Electrochem. Solid-State Lett.* **2010**, *13*, G54.
[36] S. Balatti, S. Larentis, D. C. Gilmer, D. Ielmini, *Adv. Mater.* **2013**, *25*, 1474.
[37] S. Privitera, G. Bersuker, S. Lombardo, C. Bongiorno, D. C. Gilmer, *Solid. State. Electron.* **2015**, *111*, 161.

[38] S. Yu, H.-Y. Chen, B. Gao, J. Kang, H.-S. P. Wong, *ACS Nano* **2013**, *7*, 2320.
[39] W. R. Hiatt, T. W. Hickmott, *Appl. Phys. Lett.* **1965**, *6*, 106.
[40] M. D. Pickett, G. Medeiros-Ribeiro, R. S. Williams, *Nat. Mater.* **2013**, *12*, 114.
[41] D. S. Jeong, H. Schroeder, R. Waser, *Phys. Rev. B* **2009**, *79*, 195317.
[42] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, R. S. Williams, *Nat. Nanotechnol.* **2008**, *3*, 429.
[43] M.-J. Lee, C. B. Lee, D. Lee, S. R. Lee, M. Chang, J. H. Hur, Y.-B. Kim, C.-J. Kim, D. H. Seo, S. Seo, U. I. Chung, *Nat. Mater.* **2011**, *10*, 625.
[44] F. Miao, J. P. Strachan, J. J. Yang, M.-X. Zhang, I. Goldfarb, A. C. Torrezan, P. Eschbach, R. D. Kelley, G. Medeiros-Ribeiro, R. S. Williams, *Adv. Mater.* **2011**, *23*, 5633.
[45] R. Soni, A. Petraru, P. Meuffels, O. Vavra, M. Ziegler, S. K. Kim, D. S. Jeong, N. A. Pertsev, H. Kohlstedt, *Nat. Commun.* **2014**, *5*, 5414.
[46] B. J. Choi, J. Zhang, K. Norris, G. Gibson, K. M. Kim, W. Jackson, M.-X. M. Zhang, Z. Li, J. J. Yang, R. S. Williams, *Adv. Mater.* **2016**, *28*, 356.
[47] P. Jain, U. Arslan, M. Sekhar, B. C. Lin, L. Wei, T. Sahu, J. Alzate-vinasco, A. Vangapaty, M. Meterelliyoz, N. Strutt, A. B. Chen, P. Hentges, P. A. Quintero, C. Connor, O. Golonzka, K. Fischer, F. Hamzaoglu, IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco, CA, **2019**, pp. 212–214.
[48] W. Zhao, M. Moreau, E. Deng, Y. Zhang, J.-M. Portal, J.-O. Klein, M. Bocquet, H. Aziza, D. Deleruyelle, C. Muller, D. Querlioz, N. B. Romdhane, D. Ravelosona, C. Chappert, *IEEE Trans. Circuits Syst. I Regul. Pap.* **2013**, *61*, 443.
[49] N. Xu, K. J. Yoon, K. M. Kim, L. Fang, C. S. Hwang, *Adv. Electron. Mater.* **2018**, *4*, 1800189.
[50] N. Xu, L. Fang, K. M. Kim, C. S. Hwang, *Phys. Status Solidi (RRL)* **2019**, *13*, 1900033.
[51] B. Li, B. Yan, H. Li, *Proc. 2019 Gt.* Lakes Symp. VLSI, **2019**, ACM, Tysons Corner, VA, pp. 381–386.
[52] K. M. Kim, N. Xu, X. Shao, K. J. Yoon, H. J. Kim, R. S. Williams, C. S. Hwang, *Phys. Status Solidi (RRL)* **2019**, *13*, 1800629.
[53] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, R. S. Williams, *Nature* **2010**, *464*, 873.
[54] P. Huang, J. Kang, Y. Zhao, S. Chen, R. Han, Z. Zhou, Z. Chen, W. Ma, M. Li, L. Liu, X. Liu, *Adv. Mater.* **2016**, *28*, 9758.
[55] S. Balatti, S. Ambrogio, D. Ielmini, *IEEE Trans. Electron Devices* **2015**, *62*, 1831.
[56] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, U. C. Weiser, *IEEE Trans. Circuits Syst. II Express Briefs* **2014**, *61*, 895.
[57] S. Balatti, S. Ambrogio, D. Ielmini, *IEEE Trans. Electron Devices* **2015**, *62*, 1839.
[58] M. Hu, H. Li, Q. Wu, G. S. Rose, Des. Autom. Conf. **2012**, ACM, San Francisco pp. 498–503.
[59] M. Prezioso, F. M. Bayat, B. Hoskins, K. Likharev, D. Strukov, *Sci. Rep.* **2016**, *6*, 21331.
[60] M. Pedro, J. Martin-Martinez, E. Miranda, R. Rodriguez, M. Nafria, M. B. Gonzalez, F. Campabadal, IEEE Int. Reliab. Phys. Symp., IEEE, Burlingame, CA, **2018**, p. P–CR.
[61] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, *Adv. Mater.* **2018**, *30*, 1705914.
[62] B. Yan, C. Liu, X. Liu, Y. Chen, H. Li, IEEE Int. Electron Devices Meet, IEEE, San Francisco, CA, **2017**, pp. 11–14.
[63] C. Yakopcic, M. Z. Alom, T. M. Taha, Int. Jt. Conf. Neural Networks, IEEE, Anchorage, AK, **2016**, pp. 963–970.
[64] C. Li, Z. Wang, M. Rao, D. Belkin, W. Song, H. Jiang, P. Yan, Y. Li, P. Lin, M. Hu, N. Ge, *Nat. Mach. Intell.* **2019**, *1*, 49.
[65] L. Gao, P.-Y. Chen, S. Yu, *IEEE Electron Device Lett.* **2016**, *37*, 870.
[66] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, *Nat. Electron.* **2018**, *1*, 52.
[67] R. Mochida, K. Kouno, Y. Hayata, M. Nakayama, T. Ono, H. Suwa, R. Yasuhara, K. Katayama, T. Mikawa, Y. Gohou, IEEE Symp. VLSI Technol. **2018**, IEEE, Honolulu, HI, pp. 175–176.
[68] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-W. Chang, T.-C. Chang, T.-Y. Huang, H.-Y. Kao, S.-Y. Wei, Y.-C. Chiu, C.-Y. Lee, C.-C. Lo, Y.-C. King, C.-J. Lin, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, M.-F. Chang, IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco, CA, **2019**, pp. 388–390.
[69] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang, T.-H. Hsu, Y.-C. King, C.-J. Lin, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, M.-F. Chang, IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco, CA, **2018**, pp. 494–496.
[70] F. Su, W.-H. Chen, L. Xia, C.-P. Lo, T. Tang, Z. Wang, K.-H. Hsu, M. Cheng, J.-Y. Li, Y. Xie, Y. Wang, M.-F. Chang, H. Yang, Y. Liu, IEEE Symp. VLSI Technol., IEEE, Kyoto, **2017**, pp. T260–T261.
[71] H.-J. Yoo, IEEE Int. Solid-State Circuits Conf. **2019**, IEEE, San Francisco pp. 20–26.
[72] P.-Y. Chen, B. Lin, I.-T. Wang, T.-H. Hou, J. Ye, S. Vrudhula, J. Seo, Y. Cao, S. Yu, IEEE/ACM Int. Conf. Comput. Des. IEEE Press, Piscataway, NJ, **2015**, pp. 194–199.
[73] S. Choi, J. H. Shin, J. Lee, P. Sheridan, W. D. Lu, *Nano Lett.* **2017**, *17*, 3113.
[74] S. Lashkare, N. Panwar, P. Kumbhare, B. Das, U. Ganguly, *IEEE Electron Device Lett.* **2017**, *38*, 1212.
[75] H. Tan, S. Majumdar, Q. Qin, J. Lahtinen, S. van Dijken, *Adv. Intell. Syst.* **2019**, *1*, 1900036
[76] M. Imani, Y. Kim, T. Worley, S. Gupta, T. Rosing, Des. Autom. Test Eur. Conf. Exhib., IEEE, Florence, **2019**, pp. 1591–1594.
[77] I.-T. Wang, C.-C. Chang, L.-W. Chiu, T. Chou, T.-H. Hou, *Nanotechnology* **2016**, *27*, 365204.
[78] G. C. Adam, B. D. Hoskins, M. Prezioso, F. Merrikh-Bayat, B. Chakrabarti, D. B. Strukov, *IEEE Trans. Electron Devices* **2016**, *64*, 312.
[79] H. Li, K.-S. Li, C.-H. Lin, J.-L. Hsu, W.-C. Chiu, M.-C. Chen, T.-T. Wu, J. Sohn, S. B. Eryilmaz, J.-M. Shieh, W.-K. Yeh, H.-S. Philip Wong, IEEE Symp. VLSI Technol., IEEE, Honolulu, HI, **2016**, pp. 1–2.
[80] S. Yu, Z. Li, P.-Y. Chen, H. Wu, B. Gao, D. Wang, W. Wu, H. Qian, IEEE Int. Electron Devices Meet, IEEE, San Francisco, CA, **2016**, pp. 12–16.
[81] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *Proc. IEEE*, **1998**, *86*, 2278.
[82] S. Li, N. Xiao, P. Wang, G. Sun, X. Wang, Y. Chen, H. H. Li, J. Cong, T. Zhang, *IEEE Trans. Comput.* **2018**, *68*, 239.
[83] A. M. Hassan, C. Yang, C. Liu, H. H. Li, Y. Chen, Des. Autom. Test Eur. Conf. Exhib., ACM, Lausanne, **2017**, pp. 776–781.
[84] S. Park, H. Kim, M. Choo, J. Noh, A. Sheri, S. Jung, K. Seo, J. Park, S. Kim, W. Lee, J. Shin, D. Lee, G. Choi, J. Woo, E. Cha, J. Jang, C. Park, M. Jeon, B. Lee, B. H. Lee, H. Hwang, IEEE Int. Electron Devices Meet, IEEE, San Francisco, CA, **2012**, pp. 10–12.
[85] M. Imani, Y. Kim, S. Riazi, J. Merssely, P. Liu, F. Koushanfar, T. Rosing, *Cloud Comput. (CLOUD)*, IEEE, Milan **2019**.
[86] M. Imani, A. Rahimi, D. Kong, T. Rosing, J. M. Rabaey, IEEE Int. Symp. High Perform. Comput. Archit., IEEE, Austin, TX, **2017**, pp. 445–456.

[87] T. F. Wu, H. Li, P.-C. Huang, A. Rahimi, J. M. Rabaey, H.-S. P. Wong, M. M. Shulaker, S. Mitra, IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco **2018**, pp. 492–494.
[88] L. Zheng, S. Shin, S. Lloyd, M. Gokhale, K. Kim, S.-M. Kang, IEEE Int. Symp. Circuits Syst., IEEE, Montreal, **2016**, pp. 1382–1385.
[89] B. Chakrabarti, M. A. Lastras-Montaño, G. Adam, M. Prezioso, B. Hoskins, M. Payvand, A. Madhavan, A. Ghofrani, L. Theogarajan, K.-T. Cheng, D. B. Strukov, *Sci. Rep.* **2017**, *7*, 42429.
[90] Q. Luo, X. Xu, H. Liu, H. Lv, T. Gong, S. Long, Q. Liu, H. Sun, W. Banerjee, L. Li, J. Gao, *Nanoscale* **2016**, *8*, 15629.
[91] B. Gao, Y. Bi, H.-Y. Chen, R. Liu, P. Huang, B. Chen, L. Liu, X. Liu, S. Yu, H.-S. P. Wong, J. Kang, *ACS Nano* **2014**, *8*, 6998.
[92] W. Hwang, M. M. S. Aly, Y. H. Malviya, M. Gao, T. F. Wu, C. Kozyrakis, H.-S. P. Wong, S. Mitra, Int. Conf. Hardware/Software Codesign Syst. Synth. (CODES+ISSS), ACM, Seoul, **2017**, pp. 1–2.
[93] M. Yu, Y. Cai, Z. Wang, Y. Fang, Y. Liu, Z. Yu, Y. Pan, Z. Zhang, J. Tan, X. Yang, M. Li, R. Huang, *Sci. Rep.* **2016**, *6*, 21020.
[94] P. Sun, N. Lu, L. Li, Y. Li, H. Wang, H. Lv, Q. Liu, S. Long, S. Liu, M. Liu, *Sci. Rep.* **2015**, *5*, 13504.
[95] P. W. Hollis, J. J. Paulos, *IEEE J. Solid-State Circuits* **1990**, *25*, 849.
[96] P. Häfliger, M. Mahowald, L. Watts, *Advanced in Neural Information Processing System*, Neural Information Processing Systems Foundation, Denver **1997**, pp. 692–698.
[97] A. Joubert, B. Belhadj, R. Héliot, Int. New Circuits Syst. Conf., IEEE, Seoul, **2011**, pp. 9–12.
[98] J. Schemmel, J. Fieres, K. Meier, IEEE Int. Jt. Conf. Neural Networks (IEEE World Congr. Comput. Intell., IEEE, Hong Kong, **2008**, pp. 431–438.
[99] S.-I. Amarimber, *IEEE Trans. Syst. Man. Cybern.* **1972**, 643.
[100] Z. Wang, S. Joshi, S. Savel'ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo, H. Jiang, P. Lin, C. Li, J. H. Yoon, N. K. Upadhyay, J. Zhang, M. Hu, J. P. Strachan, M. Barnell, Q. Wu, H. Wu, R. S. Williams, Q. Xia, J. J. Yang, *Nat. Electron.* **2018**, *1*, 137.
[101] S. Kumar, J. P. Strachan, R. S. Williams, *Nature* **2017**, *548*, 318.
[102] B. Yan, X. Cao, H. (Helen) Li, Des. Autom. Conf., San Francisco **2018**.
[103] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, G. Indiveri, *Front. Neurosci.* **2015**, *9*, 141.
[104] L. F. Abbott, T. B. Kepler, *Stat. Mech. Neural Networks*, Springer, Berlin, Heidelberg **1990**, pp. 5–18.
[105] L. O. Chua, *Int. J. Bifurc. chaos* **2005**, *15*, 3435.
[106] L. Chua, V. Sbitnev, H. Kim, *Int. J. Bifurc. Chaos* **2012**, *22*, 1230011.
[107] Y. Wang, W. Wen, L. Song, H. H. Li, Asia South Pacific Des. Autom. Conf., IEEE, Chiba, **2017**, pp. 776–781.
[108] X. Peng, S. Yu, IEEE Asia Pacific Conf. Circuits Syst., IEEE, Chengdu, **2018**, pp. 378–381.
[109] W. H. Chen, K. X. Li, W. Y. Lin, K. H. Hsu, P. Y. Li, C. H. Yang, C. X. Xue, E. Y. Yang, Y. K. Chen, Y. S. Chang, T. H. Hsu, Y. C. King, C. J. Lin, R. S. Liu, C. C. Hsieh, K. T. Tang, M. F. Chang, in IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco **2018**, Vol. *61*, p. 494.
[110] M. Courbariaux, Y. Bengio, J.-P. David, *Adv. Neural Inf. Process. Syst.* **2015**, 3123.
[111] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Eur. Conf. Comput. Vis., Springer, Cham, **2016**, pp. 525–542.
[112] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, Miami, FL, **2009**, pp. 248–255.
[113] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, ACM/IEEE Int. Symp. Comput. Archit., IEEE, Toronto, Ontario **2017**, pp. 1–12.
[114] B. Yan, Q. Yang, W. H. Chen, K. T. Chang, J. W. Su, C. H. Hsu, S. H. Li, H. Y. Lee, S. S. Sheu, M. S. Ho, Q. Wu M. F. Chang, Y. Chen, H. Li, IEEE Symp. VLSI Technol, IEEE, Kyoto **2019**, p. T86.
[115] S. Shukla, B. Fleischer, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky, N. Cao, C.-Y. Chen, P. Chuang, T. Fox, G. Gristede, M. Guillorn, H. Haynie, M. Klaiber, D. Lee, S.-H. Lo, G. Maier, M. Scheuermann, S. Venkataramani, C. Vezyrtzis, N. Wang, F. Yee, C. Zhou, P.-F. Lu, B. Curran, L. Chang, K. Gopalakrishnan, *IEEE Solid-State Circuits Lett.* **2018**, *1*, 217.
[116] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, V. Srikumar, *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 14.
[117] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, Y. Xie, ACM SIGARCH Comput. Archit. News, **2016**, pp. 27–39.
[118] L. Song, X. Qian, H. Li, Y. Chen, IEEE Int. Symp. High Perform. Comput. Archit., IEEE, Vienna, **2017**, pp. 541–552.
[119] X. Qiao, X. Cao, H. Yang, L. Song, H. Li, Proc. 55th Annu. Des. Autom. Conf., IEEE, San Francisco, CA, **2018**, p. 103.
[120] F. Chen, L. Song, H. H. Li, Y. Chen, Des. Autom. Conf., IEEE, Las Vegas **2019**, p. 133.
[121] H. Mao, M. Song, T. Li, Y. Dai, J. Shu, IEEE/ACM Int. Symp. Microarchitecture, IEEE, Fukuoka, **2018**, pp. 669–681.
[122] A. Ankit, I. El Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W. W. Hwu, J. P. Strachan, K. Roy, D. S. Milojicic, Proc. Twenty-Fourth Int. Conf. Archit. Support Program. Lang. Oper. Syst., ACM, Providence, RI, **2019**, pp. 715–731.
[123] Caffe2, https://caffe2.ai/ (accessed: June 2019).
[124] Pytorch, https://pytorch.org/ (accessed: June 2019).
[125] ONNX, https://onnx.ai/ (accessed: June 2019).
[126] A. Grossi, E. Nowak, C. Zambelli, C. Pellissier, S. Bernasconi, G. Cibrario, K. El Hajjam, R. Crochemore, J. F. Nodin, P. Olivo, L. Perniola, IEEE Int. Electron Devices Meet., IEEE, San Francisco, CA, **2016**, pp. 4–7.
[127] B. Yan, J. Yang, Q. Wu, Y. Chen, H. Li, IEEE/ACM Int. Conf. Comput. Des., IEEE, Irvine, CA, **2017**.
[128] C. Ho, T. Y. Shen, P. Y. Hsu, S. C. Chang, S. Y. Wen, M. H. Lin, P. K. Wang, S. C. Liao, C. S. Chou, K. M. Peng, C. M. Wu, W. H. Chang, Y. H. Chen, F. Chen, L. W. Lin, T. H. Tsai, S. F. Lim, C. J. Yang, M. H. Shieh, H. H. Liao, C. H. Lin, P. L. Pai, T. Y. Chan, Y. C. Chiao, IEEE Symp. VLSI Technol., IEEE, Honolulu, HI, **2016**, pp. 1–2.
[129] Y. Pang, B. Gao, D. Wu, S. Yi, Q. Liu, W.-H. Chen, T.-W. Chang, W.-E. Lin, X. Sun, S. Yu, H. Qian, M.-F. Chang, H. Wu, IEEE Int. Solid-State Circuits Conf. **2019**, IEEE, San Francisco, CA, pp. 402–404.
[130] S. Ambrogio, S. Balatti, V. McCaffrey, D. Wang, D. Ielmini, IEEE Int. Electron Devices Meet., IEEE, San Francisco, CA, **2014**, p. 14.
[131] P. Huang, Y. C. Xiang, Y. D. Zhao, C. Liu, B. Gao, H. Q. Wu, H. Qian, X. Y. Liu, J. F. Kang, IEEE Int. Electron Devices Meet., IEEE, San Francisco **2018**, pp. 40–44.
[132] C.-Y. Chen, H.-C. Shih, C.-W. Wu, C.-H. Lin, P.-F. Chiu, S.-S. Sheu, F. T. Chen, *IEEE Trans. Comput.* **2014**, *64*, 180.
[133] L. Chua, *IEEE Micro* **2018**, *38*, 7.
[134] C. Liu, M. Hu, J. P. Strachan, H. Li, Des. Autom. Conf., ACM, Austin, TX, **2017**, pp. 1–6.

[135] L. Xia, M. Liu, X. Ning, K. Chakrabarty, Y. Wang, Des. Autom. Conf., ACM, Austin, TX, **2017**, p. 33.
[136] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, Y. Shi, *Nat. Electron.* **2018**, *1*, 216.
[137] T. F. Wu, B. Q. Le, R. Radway, A. Bartolo, W. Hwang, S. Jeong, H. Li, P. Tandon, E. Vianello, P. Vivet, E. Nowak, IEEE Int. Solid-State Circuits Conf., IEEE, San Francisco, CA, **2019**, pp. 226–228.
[138] A. Krizhevsky, G. Hinton, Learning Multiple Layers of Features from Tiny Images, Vol. *1*, No. 4. Technical Report, University of Toronto, Toronto **2009**.
[139] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, *Int. J. Comput. Vis.* **2015**, *115*, 211.