Memristor For Computing: Myth or Reality?

Said Hamdioui[1] Shahar Kvatinsky[2] Gert Cauwenberghs[3] Lei Xie[4] Nimrod Wald[5]

Siddharth Joshi[6] Hesham Mostafa Elsayed[7] Henk Corporaal[8] Koen Bertels[9]

[1,4,9]Computer Engineering
Delft University of Technology
Delft, the Netherlands
S.Hamdioui@tudelft.nl

[2,5]Technion
Israel Institute of Technology
Haifa, Israel
shahar@ee.technion.ac.il

[3,6,7]Jacobs School of Engineering
UC San Diego,
California, USA
gert@ucsd.edu

[8]Electronic Systems group
Eindhoven Univ. of Technology
Eindhoven, The Netherlands
h.corporaal@tue.nl

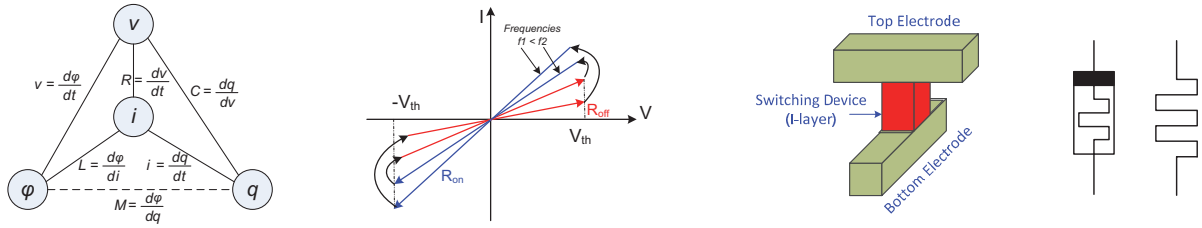
***Abstract*—CMOS technology and its sustainable scaling have been the enablers for the design and manufacturing of computer architectures that have been fuelling a wider range of applications. Today, however, both the technology and the computer architectures are suffering from serious challenges/ walls making them incapable to deliver the right computing power at pre-defined constraints. This motivates the need of exploring new architectures and new technologies; not only to maintain the economic benefit of scaling, but also to enable the solutions of emerging computer power and data storage hungry applications such as big-data and data-intensive applications. This paper discusses the emerging memristor device as complementary (or alternative) to CMOS device and shows how this device can enable new ways of computing that will at least solve the challenges of today's architectures for some applications. The paper shows not only the potential of memristor devices in enabling new memory technologies and new logic design styles, but also their potential in enabling memory intensive architectures as well as neuromorphic computing due to their unique properties such as the tight integration with CMOS and the ability to learn and adapt.**

I. INTRODUCTION

Today's and emerging applications are extremely demanding in terms of storage and computing power. Data-intensive/big-data applications and internet-of-things (IoT) will transform the future; they will not only impact the all aspects of our life, but also change a lot in the IC and computer world. Emerging applications require computing power which was typical of supercomputers a few years ago, but with constraints on size, power consumption and guaranteed response time which are typical of the embedded applications [1]. Both today's computer architectures and device technologies (used to manufacture them) are facing major challenges making them incapable to deliver the required functionalities and features. Computers are facing the three well-known walls [2]: (1) The memory wall due to the increasing gap between processor and memory speeds, and the limited memory bandwidth making the memory access the killer of performance and power for memory access dominated applications; e.g. big-data; (2) The Instruction Level parallelism (ILP) wall due to the increasing difficulty in finding enough parallelism in software/code that has to run on the parallel hardware being the mainstream today; (3) The power wall as the practical power limit for cooling is reached, meaning no further increase in CPU clock speed. On the other hand, nanoscale CMOS technology, which has been the enabler of the computing revolution, also faces three walls: (1) The Reliability wall as technology scaling leads to reduced device lifetime and higher failure rate [1], (2) The Leakage wall as the static power is becoming dominant at smaller technologies (due to volatile technology and lower Vdd) and may even be more than the dynamic power [3]; (3) The Cost wall as the cost per device via pure geometric scaling of process technology is plateauing [4]. All of these have led to the slowdown of the traditional device scaling. In order for computing systems to continue deliver sustainable benefits for the foreseeable future society, *alternative computing architectures* and notions have to be explored in the light of *emerging new device technologies*. Resistive computing, quantum computing, molecular computing, and neuromorphic computing are couple of alternative computing notions, while memristor devices, quantum dots, spin-wave devices are couple of emerging device technologies [5]. Memristor device is a promising candidate to complement and/or replace traditional CMOS (at least in some applications) due to many advantages including CMOS process compatibility, zero standby power, great scalability and high density, and its potential to implement high density memories [6-10] and different logic design styles [11-14], enabling new computing paradigms [15-21].

In this paper, we will explore the potential of memristor devices in building memories, logic functions, and new computer architectures. First the paper provides a brief overview on the memristor device, including its main properties. Second, it discusses the potential of these devices to enable the new generations of non-volatile memories as well as new generation of logic design styles. Third, the paper shows how the unique properties of memristor devices can enhance the concept of *Memory Intensive Architecture*, which makes use not only of bigger and faster on-die memories, but also of new micro-architectures and memristor devices as memory elements; two architecture examples will be demonstrated. The first uses a novel memory structure called the *multistate register* to store instruction states locally; while the second uses local nonvolatile memory elements to construct *nonvolatile processors* that can be efficiently powered-on and powered-off. Fourth, the paper discusses how the memristor offer a compelling solution for the realisation of *neuromorphic architectures*, including *non-spiking neural networks* as well *spiking (artificial) networks;* memristors encode the synaptic weights for spiking neural networks, while they encode the weight of the neurons for artificial networks. Finally, the paper highlights the major challenges for CMOS-memristor system integration.

978-3-9815370-8-6/17/$31.00 ©2017 IEEE



(a) The four fundamental elements (b) Memristor pinched hysteresis loop (c) Memristor structure (d) Symbols

Fig. 1. Main characteristics of a memristor device

II. MEMRISTOR: WHAT IS IT ABOUT?

Memristor, or more generically a memristive device, is basically a fourth class of fundamental two-terminal electrical element, joining the resistor, the capacitor, and the inductor. It was initially proposed in 1971 by the nonlinear circuit theorist Leon Chua [22]. He noticed that there is still one missing relationship between flux and charge as shown in with the dashed line in Fig.1(a). Theoretically, Memristor is a passive element that maintains a relationship between the time integrals of current and voltage across a two terminal element, while considering the internal state variable of the device; hence, a memristor can be equivalently expressed either by a function of charge *q* or a function of flux φ. The beauty of the memristor lies in its ability to remember its history (i.e., the internal state). An important fingerprint of a memristor is the 'pinched hysteresis loop' current-voltage characteristic as illustrated in Fig.1(b); for a memristor excited by a periodic signal, when the voltage *v(t)* is zero, the current *i(t)* is also zero and vice versa. Another signature of the memristor is that the 'pinched hysteresis loop' shrinks with the increase in the excitation frequency *f*. Fig.1(c) shows a typical memristor device consisting of two metallic electrodes that sandwich a thin dielectric Insulating layer (I-layer) serving as permanent storage medium making its leakage current close to zero. The exact mechanism differs significantly among the different materials being used, but the common link among all memristive devices is an electric field causes ionic movements and local structural changes in the storage medium, which in turn causes a measurable change of the device resistance. Fig.1(d) shows the two symbols typically used to denote a memristor; the black square represents the positive terminal.

It is just in 2008 that the memristor became famous, after a silent period of more than 30 years; the first physical memristor device was manufactured and demonstrated by HP Laboratory [23]. HP built a metal-insulator-metal device using titanium oxide as insulator and identified the memristive behaviour over its two-terminal node as described by Leon Chua. The device modulates its resistance by controlling positive charged oxygen vacancies in the insulator layer using different voltages. After the first memristor device was fabricated, several memristor devices based on different type of materials have been proposed such as spintronic [24], amorphous silicon [25], and ferroelectric memristors [26].

III. MEMRISTOR FOR LOGIC AND MEMORY

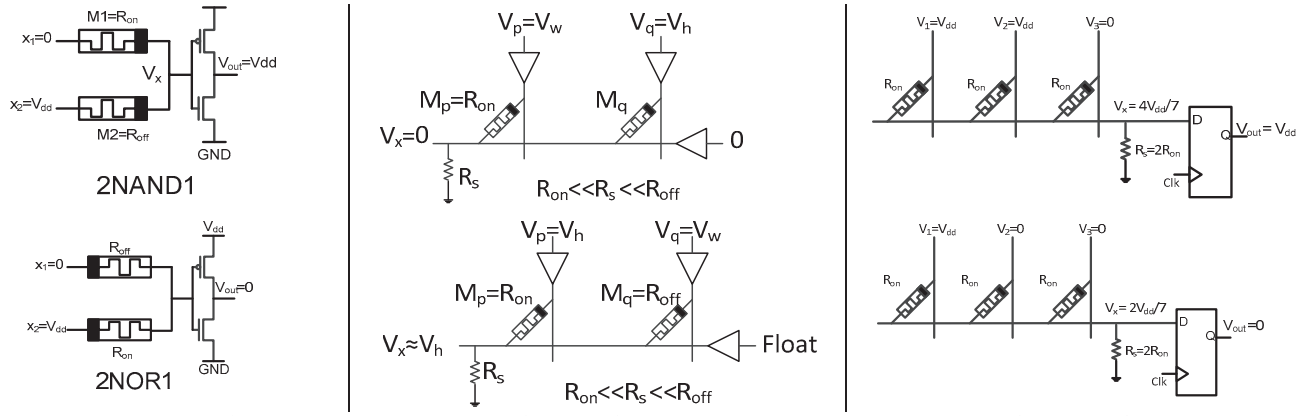
Researchers have been exploring the use of memristor devices for the implementation not only of the non-volatile memories, but also different logic styles.

A. Memristor for memories

All 2-terminal non-volatile memory devices based on *resistance switching* are *memristors*, regardless of the device material and physical operating mechanisms [9]; they all exhibit a distinctive "fingerprint" characterized by a *pinched hysteresis loop* confined to the first and the third quadrants of the *v–i* plane whose contour shape in general changes with both the amplitude and frequency of any periodic "sine-wave-like" input voltage source, or current source.

Memristive based memory devices, also referred to as resistive memory devices, are one of the most promising candidates for next generation NVM because of their faster write time, large resistance ratio R_{off}/R_{on}, and smaller write power consumption [27]. Resistive memories can be either single-level cell or multiple Multi-level cell; multi-level operation can be easily realised e.g., by modifying the current through the cell. It is expected that resistive memories will be capable of storing up to 1TB of data on a single chip, thanks to the ability of "3D-stacking" multiple cells in different configurations in order to save space while still upping the storage limits [17]. All of this can fit into a tight, tiny space, which could then fit into mobile devices and would essentially replace NAND flash memory, which is the current standard in the gadget industry, and pretty much all smartphones and tablets use it.

Resistive memory devices, can be classified based on their dominant physical operating mechanism into three classes [27]: *Phase Change Memories (PCM), Electrostatic/ Electronic Effects Memories (EEM), and Redox memories.* The physical mechanism for switching resistance states for PCM is purely thermal (e.g., an electrical power can induce Joule heating and involve the formation and disruption of some localized conduction paths in an insulating material), while that of EEM is purely based electronic effects (e.g., charge injection and trapping, ferroelectric polarization reversal, Mott metal-insulator transition). On the other hand, Redox memories are memories where the physical mechanism



(a) Memristor Ratioed Logic (b) Stateful Logic (c) Progr.CMOS/Memristor logic

Fig. 2. Examples of memristor based logic designs

for switching is based on reduction/oxidation (Redox)-related chemical effects. The category of "Redox RAM" encompasses a wide variety of *Metal-Insulator-Metal (MIM)* structures; the electrochemical mechanisms driving the resistance state (from high to low or the reverse) can operate in the bulk I-layer, along conducting filaments in the I-layer, and/or at the I-layer/metal contact interfaces in the MIM structure.

B. Memristor for logic

Several memristor based logic design have been proposed; these can typically classified, based on the nature of the logic designs, into three classes: *Boolean Logic* [12, 13, 28, 29, 30]. *Implication Logic* [31, 32, 33], and *Threshold/Majority* [34, 35]; they are briefly discusses next.

Boolean Logic

Boolean Logic provides the conventional mathematical algebra primitives such as conjunction (AND), the disjunction (OR), and the negation (NOT) applied to values of *variables*. Four different implementation families were proposed:

- CMOS-Like Memristor based logic design (CLM) [28]: This can implement any arbitrary Boolean functions by replacing the pull-up and pull-down network as used in CMOS gates with memristors. E.g., a CLM two-input NAND gate consists of two parallel memristors in the pullup network (with the positive terminals connected to Vdd), and two serial memristors in the pull-down network (with the positive terminals oriented to GND).
- Memristor Ratioed Logic (MRL) [29]: This is hybrid CMOS-memristive logic family. In this logic, OR and AND logic gates are based on memristive devices, and CMOS inverters are added to provide a complete logic structure and signal restoration. The top part of Fig. 2(a) illustrates a two input NAND gate. When input x1x2=01, the current flows from memristor M2 to M1 causing them to switch to R_{on}, respectively, R_{off}; see also Fig 1(b). As a result, the voltage Vx is close to 0 resulting in V_{out}= Vdd. A two-input NOR is shown in bottom part of Fig. 2(a); note that the polarity of the memristors is the only structural difference.
- MAGIC Logic [12]: this is a memristor-only logic family. In each logic gate, memristors serve as inputs with previously stored data, and an additional memristor serves as an output. The applicability of such logic to memristor-based crossbar is straightforward in case of NOR gate, while it requires an additional resistor within each row in case of other gates.
- Crossbar memristor based logic [13, 30]: This is also a memristor-only logic family; using appropriate controlling signals and procedure to the crossbar, inverting and non-inverting gates, and even latches and sum-of-product logic functions can be implemented. It has been shown even that any Boolean function can be implemented in seven steps irrespective of its size/minterms and complexity [13]. Hence, multiple memristor crossbars may be combined to implement complex computational systems.

Implication Logic

Implication Logic provides primitive logic operations on typically two *propositions*; examples are material implication, converse implication, and material non-implication. Two families of memristor based implication logic are proposed.

- Stateful Logic [31, 32]. In this memristive logic family, each memristor is used as an input, output, computational logic element, and latch in different stages of the computing process. The logical state is determined by the resistance of the memristor. This logic family can be integrated within a memristor-based crossbar, commonly used for memory. Fig. 2(b) illustrates the working principal for "*if p then q*" (p→q) in case *p* is TRUE and *q* is FALSE. First *p* and *q* are programmed to R_{on}, respectively, R_{off} by providing appropriate voltages to the control signals (bit lines and the wordline). E.g., top part of Fig. 2(b) shows how to program Ron by applying $V_p=V_w$ and $V_q=V_h$, where $V_w>V_{th}$ is the writing voltage (V_{th} is the threshold voltage to switch the memristor), and $V_h=V_w/2<V_{th}$ used to prevent non-selected memristors from switching. Note that the voltage across Mp is $V_w>V_{th}$; hence Mp is set to R_{on}. In a similar way, Mq can be programmed to R_{off}. Next, a computing step is executed, as shown in the bottom part of Fig. 2(b); it applies $V_p=V_h$, $V_q=V_w$, and keeps the wordline floating, resulting in $V_x\approx V_h$,

which forces the output memristor Mq to remain in its R_{off} state (i.e. FALSE).

- Complementary Resistive Switch (CRS) Logic [33]: this implementation is based on the use complementary resistive switches (i.e., two antiserial memristive elements) in order to enable the selection of designated cell(s) within a passive crossbar array without interference from sneak-path currents through neighbouring cells. Note that the CRS concept can be also used to build Boolean logic.

Threshold/Majority

Threshold logic provides threshold gates in which the output value depends on whether the arithmetic sum of values of its weighted inputs exceeds a certain threshold. Majority logic is actually a subset of threshold logic with the constraint that the inputs are all binary and the weights are all equal. These non-traditional styles enables e.g. neuromorphic computing. Three implementation families of such logic are proposed.

- Programmable CMOS/Memristor Logic [34, 35]. In this family, memristive devices implement ratioed diode-resistor logic, while CMOS circuitry is used for signal amplification and inversion; this logic is even in-field configurable. The top part of Fig. 3(c) illustrates a three-input threshold logic gate implemented with four parallel memristors connected to a pull down resistor and a CMOS D flip-flop. The three voltage inputs are first weighed by the conductance of memristors which are actually transferred to currents; these are summed up, and transferred to a voltage V_x by R_s. If $V_x > V_{th}$ of the D flip-flop, then the output is set to 1. The top part of Fig. 3(c) shows the case when the output result is 1 and the bottom part when the output is 0, assuming that $V_{th}=Vdd/2$ and $R_s=2R_{on}$ of the memristor.
- Hybrid Current Mirror Logic [34]: For this implementation, the weights of a threshold function are represented with memristance such that Ohm's Law is used to convert voltage signal inputs into weighted inputs represented by currents. The currents determined for all inputs must be summed and then compared with a reference current representing the threshold. If the summed current exceeds the threshold, the output of the gate is pulled to the high supply voltage or vice versa. This operation can be accomplished using current mirrors to first reflect the weighted inputs, sum them, and then compare to the threshold represented by the reference current.

IV. Memristor for Memory-Intensive Architectures

As already mentioned, the state-of-the art computers are facing memory and power walls. The trends in modern memory systems have been changing towards throughput and energy efficiency. Fig. 3 illustrates throughput/memory-bandwidth versus bandwidth. Obviously it is desired to achieve high throughput, but additionally not to have a large (and costly) bandwidth systems (bandwidth demon systems). *Memory Intensive Architectures* target the realization of similar overall throughput as systems with large bandwidths, but then by using small bandwidth, smart microarchitectures, increasing on-die memory and new device technologies (such as memristors).

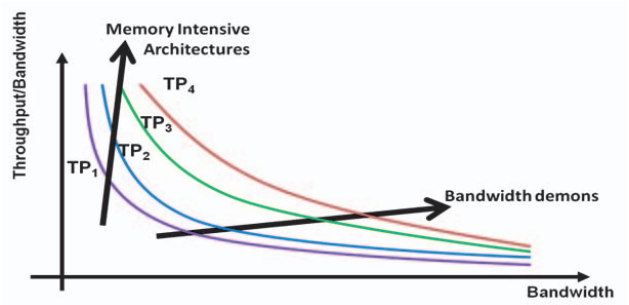


Fig. 3. Throughput/memory-bandwidth vs. bandwidth. Each curve on the graph has a fixed throughput *TPi* (performance) while the memory bandwidth varies.

The unique properties of memristive technologies, together with their speed, high endurance and low power, make them ideal candidates to enable memory intensive architectures. Memristors can be used in memory levels different than the standard non-volatile memory (*i.e.*, different than secondary storage and code storage). Additionally, many types of memristors can be fabricated between two layers of metal and can therefore be fabricated in back-end-of-the-line of any conventional CMOS process, including stacking them in multiple vertical layers. Fabricating such a tight integration, where memristors are located on-top of CMOS transistors enables extremely high dense memory with low access latency. Having numerous memory elements located above the CMOS logic is the key to enable the radical change in computer architecture that we call the *Memory Intensive Computing* era.

Memory intensive computing does not only mean bigger and faster on-die conventional memories (*e.g.*, cache, register file), but also includes new micro-architectures and novel memory elements, which use the "sea of memory" located on the top of the standard CMOS circuits, to enhance the overall performance of processors and reduce the power dissipation. These advantages of using an ample amount of on-die memory make the technology appealing for use in big data and internet of things (IoT) applications.

We demonstrate the concept of memory intensive computing by two examples. The first example uses a novel memory structure called the *multistate register* to store instruction states locally and by that significantly improving performance in multithreaded processors. The second example uses local nonvolatile memory elements to construct *nonvolatile processors* that can be efficiently powered-on and powered-off and by that, substantially reducing energy. It is worth noting that Hybrid Memristor-CMOS Logic (as explained in Section III) is another option that can be explored for memory intensive architectures; it can be used exploit the "sea of memory elements" via a tight integration on the top of CMOS logic to create three-dimensional logic paths that dramatically improve area efficiency.

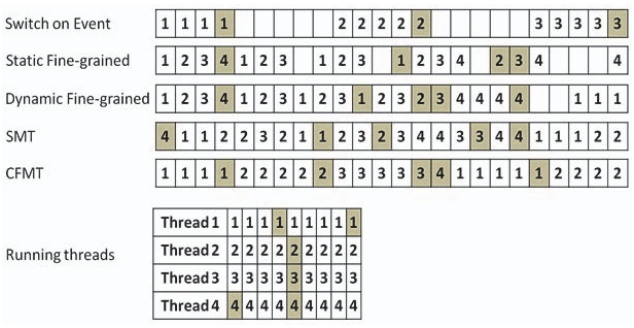


Fig. 4. Illustration of timelines for different multithreading techniques with four threads. All processors run the same four threads as shown in the lower table. The latencies of 'white' and 'shaded' instructions are, respectively, a single clock cycle and ten clock cycles.

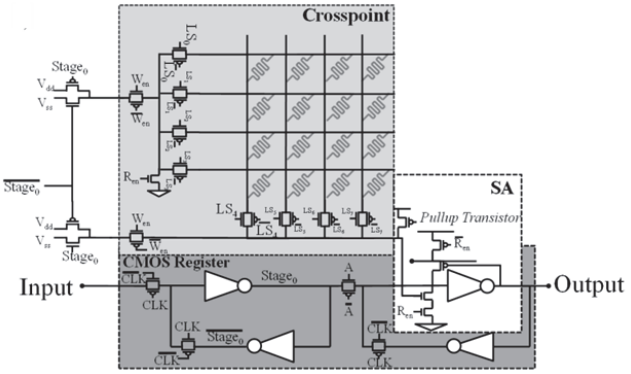


Fig. 5. Single bit RRAM-based memristive multistate register. The active bit is stored within a CMOS D flip flop and the other 15 inactive bits are stored within the memristive crossbar

A. Continuous Flow Multithreading (CFMT)

The newly added abundant memory can be used to increase the performance of a machine by increasing the capacity of elements within the processor, such as caches, branch predictors, instruction queues, prefetching structures, reorder buffers, and other buffers. The additional memory elements can also be used to store data, which currently is not being stored due to the limitations of conventional technologies. For example, it is possible to store the results of previously executed instructions to perform instruction reuse and have hardware memorization [36]. It is also possible to have many checkpoints within the processor. Another example for storing previously unsaved data is demonstrated in a novel microarchitecture named *Continuous Flow Multithreading* (CFMT) [18], which will be discussed next.

CFMT is a multithreaded microarchitecture based on Switch on Event multithreading (SoE MT) [37, 38, 39]; it employs the simple method of switching between active threads each time an event occurs that takes more than a certain amount of time to resolve (*e.g.*, cache miss). SoE MT is a simple technique that requires minimal control, but it suffers from low performance and relatively high energy consumption due to pipeline flushing on every thread switch; this is exactly what is avoided by CFMT. The superiority of CFMT comes from the fact that it can *locally* stores any relevant data such as that stored in in pipeline registers and register files; these locally stored data can be then used later on a thread switch. This avoids flushing of early pipeline stages, thus reducing thread switch penalty. Fig. 4 Illustrates the instruction flow of CFMT compared to other multithreading methods including SoE, Fine-grained [40, 41], and Simultaneous Multi-Threading (SMT) [42]. In addition, the reduced penalty allows to consider new events as worthy of a thread switch, further increasing throughput.

The primary change in CFMT as compared to SoE MT is the use of a novel memristive memory element named *Multistate Pipeline Register* (MPR) [43]. The MPR replaces the conventional pipeline register and is used to store the states of instructions from different threads. The MPR has a CMOS layer that is similar to conventional pipeline register and a memristive layer that is used to store additional data, as illustrated for a resistive memory (RRAM) based memristive MPR in Fig. 5. Rather than flushing an instruction during a thread switch, the state of the instruction is stored in the memristive layer of the MPR (e.g., crosspoint in the figure), while the instruction from an active thread is stored in the CMOS layer (e.g. CMOS register in the figure). The use of memristors to store additional data is extremely area efficient. For example, in RRAM-based memristive MPR, the area required to store 64 inactive threads in the memristive layer does not exceed 86% of that required to store the active thread in the CMOS layer [44].

In addition to area efficiency, CFMT processors achieve high performance, while maintaining the control complexity as low as SoE MT, resulting in lower energy. CFMT outperforms SoE by 40% on average, and has comparable performance to SMT [44]. The CFMT example clearly highlights how the density and tight integration of memristors and CMOS makes the use of MPR area efficient, and as a result enables a multithreaded processor that achieves high performance, while keeping the energy low.

B. Low Power Nonvolatile Processors(NVP)

Similarly to MPR, *non-volatile flip-flops* (NVFFs) [45, 46] can be used to reduce energy of processors. While MPR holds multiple states, the NVFF holds a single state to store the entire state of the machine in a non-volatile memory prior to power loss. The stored state creates a "checkpoint" to restore from upon power resumption. This type of system is called a *Non-Volatile Processor* (NVP) [47, 48], and is useful in extremely low-power applications, such as IoT applications [49], where devices are isolated, and often have to rely on self-produced (harvested) energy. Such devices (*e.g.* sensors, wearables, smart IDs) are likely to spend most of the time turned off, with only brief periods of battery powered operation. The need for a standard boot from non-volatile storage before each operational period is too energy consuming and may not be feasible when using harvested energy.

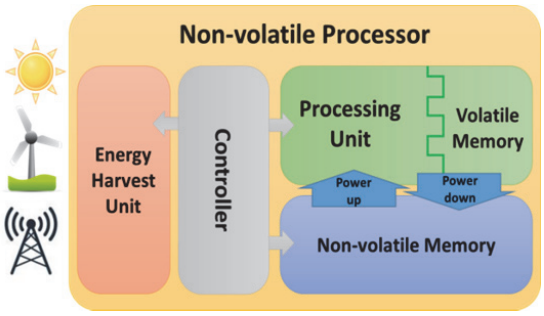


Fig. 6: Basic structure of a Non-Volatile Processor NVP

Recent reported NVFFs have read/write latencies in the order of 1's to 100's of nanoseconds, and energy dissipation of few pJ per bit [50, 51, 52, 53]. These properties, combined with the tight integration of CMOS and memristors, make memristors ideal candidates for local storage of the machine state for resuming operation after a power down period. In fact, low latency and energy costs allow storing a device's state not only before a planned power down, but also in the case of a sudden power failure.

Fig. 6 shows the structure of an NVP; the processor consists of an energy module for harvesting and managing an ambient energy source (*e.g.*, solar, wind, thermal, piezoelectric, RF), a (typically simple) processing unit with volatile memory (FFs, caches, working set memory), and a non-volatile part for check-pointing. These are all controlled by a central controller that determines the mode of operation. While using conventional non-volatile technologies compel the non-volatile part to be a separate module, when using memristive NVFF, the nonvolatile part can be fabricated directly above the CMOS area, reducing wire distances for backup of data, thus decreasing latency and capacitance.

V. Memristor for Neuromorphic Computing

Biological neural networks process information in a qualitatively different manner from conventional digital processors. Unlike the sequence of instructions programming model employed by conventional von Neumann architectures, the knowledge, or the program, in a neural network is largely encoded in the pattern and strength/weight of the synaptic connections. This "programming model" is key to the adaptability and resilience of neural networks. That is because through synaptic plasticity mechanisms, the network can continuously learn, or self-program, by adjusting the weights of the synaptic connections based on the network inputs and internal dynamics.

Neuromorphic computing draws inspiration from biological networks in an effort to develop adaptable and massively parallel computing architectures. In biological networks, synapses outnumber neurons by several orders of magnitude, making the implementation of plastic synaptic elements the key hurdle limiting the scalability of neuromorphic architectures. Memristors offer a compelling solution to the scalability problem, as they can be used as nano-scale synaptic elements whose connection weights (conductances) are modulated by the activity of the pre- and post-synaptic neurons, as is the case in biological networks.

Neural network architectures that make use of memristive devices roughly fall into two categories: architectures based on *spiking neural networks* that make use of biologically motivated spike-base plasticity mechanisms, and architectures based on *non-spiking, or artificial, neural networks* where connection weights are updated according to some form of gradient descent. Artificial neural networks have been tremendously successful in solving a wide class of machine learning problems. However, they make use of analog-valued neurons and are only loosely related to biological networks. While not neuromorphic in the full sense of the word, artificial neural networks still embody the main principles of operation of biological networks such as the distributed representation of information in neural activity, and adaptability through synaptic plasticity.

Neuromorphic architectures place less stringent demands on the memristor characteristics compared to conventional digital circuits. Learning in artificial networks is an iterative process that adjusts the connection weights based on the network behavior. This iterative process can compensate for mismatch in the memristive synaptic elements, as well as unreliable set/reset mechanisms in the memristive device. Unreliable set/reset mechanisms can even be beneficial, as intermittent failures in updating the synaptic weights can make learning more robust against spurious inputs [54]. Like their biological counterparts, some artificial network models can even tolerate the complete failure of some synaptic elements, as the synaptic learning mechanisms can use the remaining connections to compensate [55]. The required synaptic weight resolution will depend on the network and the learning task. Several recent results indicate that the 32-bit floating point weights typically used in artificial neural network models are far from necessary and synaptic bit-resolutions as low as 4-bit [56], 2-bit [57], or even 1-bit are sufficient [58] during forward propagation.

In the rest of this section we will show how the memristor enable the implementation of artificial neural network and spiking neural network architectures.

A. Memristors in artificial neural network architectures

Artificial neural networks (ANNs) utilize analog valued neurons. One common ANN architecture is the feedforward fully connected network. These networks are composed of successive layers of neurons with feedforward connections from one layer to the next and no lateral connections within layers. The output of the neurons at layer *l*, y^l is given by:

$$y^l = \sigma(\boldsymbol{W^l} y^{l-1} r + b^l)$$

where $\mathbf{W^l}$ is the matrix of weights connecting neurons in layer *l-1* to neurons in layer *l* and b^l is the bias vector for the neurons in layer *l*. σ is a static non-linearity. The first (bottom) layer is the input layer carrying in the raw data to be processed.

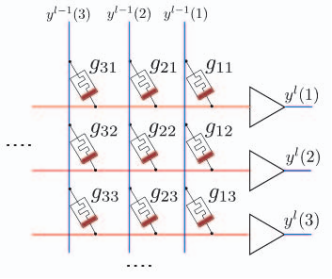
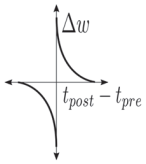
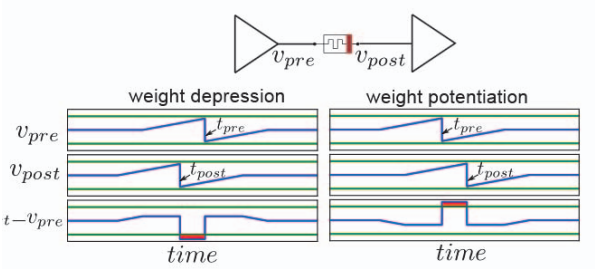


Fig. 7. General form of a memristive ANN implementation of a single feedforward layer. The input layer neural activity, y^{l-1}, is encoded as analog voltages. The output neurons maintain a virtual ground at their input terminals and their input currents represent weighted sums of the activities of the neurons in the previous layer, where the weights are encoded in the memristors' conductances. The output neurons generate an output voltage proportional to their input currents (after applying the static non-linearity, σ).



(a)



(b)

Fig. 8. (a) General shape of the STDP curve illustrating the relation between weight change and relative timing between pre- and post-synaptic spikes, $t_{post} - t_{pre}$. (b) STDP learning implemented using a memristor with bipolar switching characteristics. The waveforms generated on the pre- and post-synaptic terminals cause the voltage applied across the memristor, $v_{post} - v_{pre}$, to cross the conductance modulation threshold only when the pre- and post-synaptic spikes are temporally close. The green horizontal lines indicate the modulation threshold. The polarity, amplitude, and duration of the threshold-crossing depends on $t_{post} - t_{pre}$. as illustrated here for the weight potentiation and depression cases.

A straightforward memristive realization is illustrated in Fig. 7. Fully connected feed-forward networks lend themselves very well to implementation using memristor crossbar arrays. A single layer network based on the architecture in Fig. 7 has been physically implemented using Titanium dioxide memristors [59].

B. Memristors in spiking neural network architectures

In memristor-based spiking neural networks, the conductance of the memristor is used to encode the synaptic weight. Neurons are either implemented using standard CMOS circuits, or logically implemented in software on a digital processor. Synaptic plasticity in spiking neural networks often takes the form of spike-timing dependent plasticity (STDP) where the synaptic weight is updated based on the relative difference between the spike times of the pre- and post-synaptic neurons. STDP typically induces weight increase (decrease) when the pre-synaptic spike precedes (lags) the post-synaptic spike as shown in Fig 8(a). This results in a Hebbian learning mechanism that strengthens the connection between neurons whose activities are causally related. Implementations of STDP typically exploit the threshold effects observed in the switching characteristics of several types of memristors [60], which cause applied voltages below a threshold to have little to no effect on memristor conductance. STDP-driven weight updates can be realized by applying a shaped pulse to the memristor's pre-(post-)synaptic terminal whenever the pre-(post-) synaptic neuron spikes. The pulses are shaped so as to induce a change in memristor conductance only when the pre- and post-synaptic spikes occur in close temporal proximity as shown in Fig. 8(b) [61]. Learning rules based on STDP have been used to train spiking networks on image classification tasks [62] and on probabilistic inference tasks [63]. The later network model has recently been physically implemented using Titanium dioxide memristors as synaptic elements [64].

Several biologically motivated learning rules do not fall into the vanilla STDP learning paradigm where weight update is only a function of the time difference between pre- and post-synaptic pikes [65, 66]. These learning rules usually employ extra state variables in the neurons that modulate the magnitude of the weight updates. The pulse shaping paradigm illustrated in Fig. 8(b) can still be applied to implement some of these rules by modulating the shape of the spike-triggered waveforms based on the plasticity-relevant internal states of the neuron [67], which would in turn allow these states to control the magnitude of the weight update. Some learning rules, however, trigger synaptic updates on pre-synaptic spikes [54], even in the absence of post-synaptic spikes. These rules therefore cannot depend on the overlap between pre- and post-synaptic waveforms to trigger memristor weight updates. An alternative stimulation protocol can be used in which the memristor pre-synaptic terminal is kept floating and then clamped to an intermediate voltage on pre-synaptic spikes. The post-synaptic terminal potential would then control the magnitude/polarity of the weight update based on the post-synaptic neuron's state [68]. This, however, precludes the use of memristor cross-bar arrays since the floating pre-synaptic terminals would then introduce direct paths involving two memristors in series between the post-synaptic terminals. A generalization of this stimulation scheme to crossbar memristor arrays [69] uses active sensing circuits on the post-synaptic side which detect and lock onto the pre-synaptic stimulation pulse, and drive a post-synaptic stimulation pulse in sync with the pre-synaptic pulse. All memristor terminals are actively driven at all times, thereby avoiding direct current paths between post-synaptic terminals which could be mistakenly interpreted as pre-synaptic activity.

Importantly, the inherent robustness of neuromorphic architectures for general tasks in statistical inference and learning make them resilient to imprecision in their physical implementation using highly variable and noisy memristor devices. The stochasticity of ionic transport in memristor junctions can even be harnessed to implement stochastic algorithms in neural transduction [19] and in synaptic transmission [70, 71] yielding greater performance in learning

and inference, in addition to greater energy efficiency and noise resilience.

VI. CMOS-MEMRISTOR SYSTEM INTEGRATION CHALLENGES

Despite their many advantages, several open problems and challenges remain in integrating memristive devices with conventional CMOS systems.

Modern CMOS has curbed scaling induced power dissipation through lowering operating voltages below 1V for the core computational circuitry [72]. However, the physics of resistive memory devices necessitates energy be expended to read and write to them [26, 27, 73]. Furthermore, fabricating and *electroforming* these devices often entails high electric fields being applied across the device terminals to SET the device. Analysis of the scaling properties of these devices shows the area of the device scales inversely with the intensity of the forming voltage [74] with voltages >2V required to form $100nm^2$ HfO_x based memristors [75]. The forming process thus requires the inclusion of high-voltage devices in the periphery reducing memory densities. These voltages are generally determined by the high-resistance state (HRS), the desired low-resistance state (LRS) and the set-compliance current which prevents dielectric breakdown during the forming process. Attempts at reducing this voltage lead to an increase in the LRS, and, decreasing R_{HRS}/R_{LRS} leading to indistinguishable states; see also Fig. 1(b). This effect is further exacerbated when the selector device in a 1T1R structure is replaced by a *vertical* diode (1D1R) memory cell structure [76] due to further lack of control on the compliance current.

Challenges in scaling up the size of arrays of memristors include: nonuniform resistance profile across the crossbar array, resistance drift, inherent device-to-device and cycle-to-cycle variations as well as yield issues. Overcoming these requires a concerted effort at both a device and algorithmic level. While highly fault-tolerant algorithms can partly compensate for variation trading-off energy and time for accuracy [77], these need to be complemented by electrical methods like correcting pulses to restore state after reading, mitigating the effect of resistance [78]. Other techniques where memristive elements are implemented as complementary resistive switches [79]; adaptive writing and erasing methods [80] are adopted to improve yield, combating reliability issues.

The large *fan-out* and *fan-in* in contemporary machine-learning algorithms also poses specific challenges, necessitating large memory array sizes. For large arrays, the energy to drive the array dominates the net energy [81], while also suffering from increased IR drops along bit and word lines. Increasing the wire thickness to mitigate the effect of the IR drops results in decreased device densities, increased capacitance and increased energy losses in driving the array. Alternative architectures composed of adequately smaller sized arrays strike a balance between greater fill density in connectivity and more frequent communication between cores, which may lead to substantial system-level energy savings [82]. Greatest benefits of system-level integration can be reaped from hybrid CMOS-memristor architectures as tiled arrays of neural processor cores with locally dense memristor cross-bar synaptic connectivity within each core, and with globally sparse, dynamically reconfigurable axonal connectivity and event-driven neural communication across cores, as highlighted in Fig. 9. The combination of high energy efficiency and great functional flexibility of such reconfigurable architectures opens the door to system-level co-optimization of advances in memristor devices and deep learning algorithms leading to even greater advances in adaptive computational intelligence.

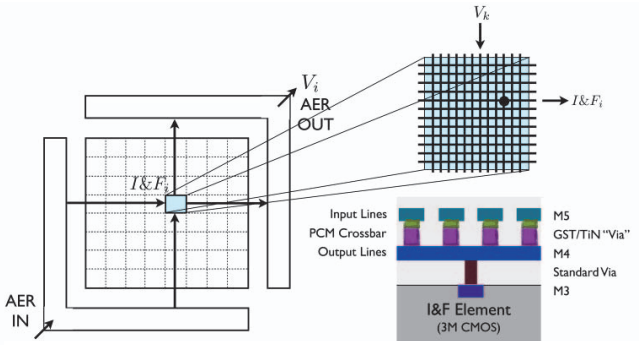


Fig. 9. Hybridization and nanoscale integration of CMOS neural arrays with synapse crossbar arrays. CMOS integrate-and-fire array transceiver (IFAT) **[83]**vertically interfacing with a nanoscale phase-change memory (PCM) synapse crossbar at greater pitch density defined by synaptic fan-in/out. The integration of IFAT neural and PCM synapse arrays externally interfacing with hierarchical event-routing (HiAER) neural event communication combines the advantages of highly flexible and reconfigurable HiAER-IFAT neural computation and long-range connectivity **[85, 84]** with highly energy-efficient local synaptic transmission.

VII. CONCLUSION

Memristive devices provide many opportunities; not only to enable new generations of non-volatile memories (with fast access speed, high reliability, up to 1TB storage capacity, and multi-level capability), but also to enable new alternative computing architectures and notions required for emerging applications; examples are smart microarchitectures making use of memristor devices as local storage elements (Memory Intensive Architectures) and neuromorphic computing making use of the memristors as weighted elements. However, there are still many challenges ahead to be solved, such as those related to CMOS-memristor system integration.

REFERENCES

[1] EU-Commission, "Next Generation Computing Roadmap, A study prepared for the European Commission," European Union, 2014.

[2] D. A. Patterson, "Future of computer architecture," Berkeley EECS Annual Research Symposium (BEARS), College of Engineering, UC Berkeley, US, 2006.

[3] B. Hoefflinger, "Chips 2020: A Guide to the Future of Nanoelectronics',," in *The Frontiers Collection, Springer Berlin Heidelberg, pp. 421-427*, 2012.

[4] H. Jones, "Whitepaper: Semiconductor Industry from 2015 to 2025," International Business Strategies, 2015.

[5] ITRS, "Beyond CMOS White Paper," ITRS, 2014.

[6] M. F. Chang, A. Lee, P. C. Chen, C. J. Lin, Y. C. King, S. S. Sheu and T. K. Ku, "Challenges and Circuit Techniques for Energy-Efficient On-Chip Nonvolatile Memory Using Memristive Devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems,* vol. 5, no. 2, pp. 183-193, 2015.

[7] A. Kawahara, R. Azuma, Y. Ikeda, K. Kawai, Y. Katoh, Y. Hayakawa and T. Takagi, "An 8Mb multi-layered cross-point ReRAM macro with 443MB/s write throughput," *IEEE Journal of Solid-State Circuits,* vol. 48, no. 1, pp. 178-185, 2013.

[8] S. Hamdioui, H. Aziza and G. C. Sirakoulis, "Memristor based memories: Technology, design and test," *IEEE International Conference On Design & Technology of Integrated Systems In Nanoscale Era,* pp. 1-7, 2014.

[9] L. Chua, "Resistance Switching Memories Are Memristors," in *Memristor Networks*, Springer, 2014, pp. 21-51.

[10] R. Fackenthal, M. Kitagawa, W. Otsuka, K. Prall, D. Mills, K. Tsutsui and G. Hush, "A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology," *IEEE International Solid-State Circuits Conference,* pp. 338-339, 2014.

[11] W. Lu, K. H. Kim, T. Chang and S. Gaba, "Two-terminal resistive switches (memristors) for memory and logic applications," *Asia and South Pacific Design Automation Conference,* pp. 217-223, 2011.

[12] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman and U. C. Weiser, "MAGIC - Memristor-Aided Logic," *IEEE Trans. on Circuits and Systems-II,* vol. 61, no. 11, pp. 895-899, 2014.

[13] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui and K. Bertels, "Fast Boolean Logic Mapped on Memristor Crossbar," *IEEE International Conference on Computer Design,* pp. 335-342, 2015.

[14] J. Cong and B. Xiao, "FPGA-RPI: A novel FPGA architecture with rram-based programmable interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 22, no. 4, pp. 864-877, 2014.

[15] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu and Y. Xie, "Pinatubo: a processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," *Design Automation Conference,* pp. 173-178, 2016.

[16] S. Hamdioui, L. Xie, H. A. Du-Nguyen, M. Taouil, K. Bertels, H. Corporaal and J. van Lunteren, "Memristor based computation-in-memory architecture for data-intensive applications," *Design, Automation & Test in Europe,* p. 1718–1725, 2015.

[17] J. J. Yang, D. B. Strukov and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnology,* vol. 8, no. 1, pp. 13-24, 2013.

[18] S. Kvatinsky, Y. H. Nacson, Y. Etsion, E. G. Friedman, A. Kolodny and U. C. Weiser, "Memristor-based multithreading," *IEEE Computer Architecture Letters,* vol. 13, no. 1, pp. 41-44, 2014.

[19] M. Al-Shedivat, R. Naous, G. Cauwenberghs and K. Salama, "Memristors empower spiking neurons with stochasticity," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems,* vol. 5, no. 2, p. 242–253, 2015.

[20] H. A. Du Nguyen, L. Xie, M. Taouil, R. Nane, S. Hamdioui and K. Bertels, "Computation-in-memory based parallel adder," *IEEE/ACM International Symposium on Nanoscale Architectures,* pp. 57-62, 2015.

[21] A. Haron, J. Yu, R. Nane, M. Taouil, S. Hamdioui and K. Bertels, "Parallel matrix multiplication on memristor-based computation-in-memory architecture," *IEEE High Performance Computing & Simulation,* pp. 759-766, 2016.

[22] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory,* vol. 18, no. 5, p. 507–519, 1971.

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

[24] Y. V. Pershin and M. Di Ventra, "Spin memristive systems: Spin memory effects in semiconductor spintronics," *Physical Review B,* vol. 78, no. 11, p. 113309, 2008.

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

[26] A. Chanthbouala, V. Garcia, R. O. Cherifi, K. Bouzehouane, S. Fusil, X. Moya and M. Bibes, "A ferroelectric memristor," *Nature Materials,* vol. 11, no. 10, p. 860–864, 2012.

[27] A. Chen, "Electronic Effect Resistive Switching Memories," ITRS, 2010.

[28] I. Vourkas and G. C. Sirakoulis, "A novel design and modeling paradigm for memristor-based crossbar circuits," *IEEE Transactions on Nanotechnology,* vol. 11, no. 6, p. 1151–1159, 2012.

[29] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser and E. G. Friedman, "MRL-Memristor ratioed logic," *IEEE International Workshop on Cellular Nanoscale Networks and their Applications,* pp. 1-6, 2012.

[30] G. Snider, "Computing with hysteretic resistor crossbars," *Applied Physics A,* vol. 80, no. 6, p. 1165–1172, 2005.

[31] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart and R. S. Williams, "Memristive switches enable stateful logic operations via material implication," *Nature,* vol. 464, no. 7290, p. 873–876, 2010.

[32] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny and U. C. Weiser, "Memristor-based material implication (imply) logic: design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 22, no. 10, p. 2054–2066, 2014.

[33] E. Linn, R. Rosezin, S. Tappertzhofen and R. Waser, "Beyond von neumann? logic operations in passive crossbar arrays alongside memory operations," *Nanotechnology,* vol. 30, no. 305205, p. 23, 2012.

[34] G. S. Rose, J. Rajendran, H. Manem, R. Karri and R. E. Pino, "Leveraging memristive systems in the construction of digital logic circuits," *Proceedings of the IEEE,* vol. 100, no. 6, p. 2033–2049, 2012.

[35] L. Gao, F. Alibart and D. B. Strukov, "Programmable cmos/memristor threshold logic," *IEEE Transactions on Nanotechnology,* vol. 12, no. 2, p. 115–119, 2013.

[36] E. Schnarr and R. L. James, "Fast out-of-order processor simulation using memoization," *ACM SIGOPS Operating Systems Review,* vol. 32, no. 5, pp. 283-294, 1998.

[37] M. Farrens and A. R. Pleszkun, "Strategies for Achieving Improved Processor Throughput," *IEEE/ACM International Symposium on Computer Architecture,* pp. 362 - 369, 1991.

[38] R. Gabor, S. Weiss and A. Mendelson, "Fairness and throughput in switch on event multithreading," *IEEE/ACM International Symposium on Microarchitecture,* pp. 149-160, 2006.

[39] J. M. Borkenhagen, R. J. Eickemeyer, R. N. Kalla and S. R. Kunkel, "A multithreaded PowerPC processor for commercial servers," *IBM Journal of Research and Development,* vol. 44, no. 6, pp. 885-898, 2000.

[40] G. T. Byrd and M. A. Holliday, "Multithreaded Processor Architectures," *IEEE Spectrum,* vol. 32, no. 8, p. 38–46, 1995.

[41] X. E. Chen and T. M. Aamodt, "A First-Order Fine-Grained Multithreaded Throughput Model," *IEEE International Symposium on High Performance Computer Architecture,* p. 329–340, 2009.

[42] S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, R. L. Stamm and D. M. Tullsen, "Simultaneous multithreading: A platform for next-generation processors," *IEEE micro,* vol. 17, no. 5, pp. 12-19, 1997.

[43] S. Kvatinsky, Y. H. Nacson, Y. Etsion, A. Kolodny, U. C. Weiser, R. Patel and E. G. Friedman, "Memristive multistate pipeline register," *IEEE International Workshop on Cellular Nanoscale Networks and their Applications,* pp. 1-2, 2014.

[44] R. Patel, S. Kvatinsky, E. G. Friedman and A. Kolodny, "Multistate Register Based on Resistive RAM," *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems,* vol. 23, no. 9, p. 1750–1759, 2015.

[45] S. Onkaraiah, M. Reyboz, F. Clermidy, J. M. Portal, M. Bocquet, C. Muller and A. Amara, "Bipolar ReRAM based non-volatile flip-flops for low-power architectures," *IEEE New Circuits and Systems Conference,* pp. 417-420, 2012.

[46] W. Zhao, E. Belhaire and C. Chappert, "Spin-MTJ based non-volatile flip-flop," *IEEE Conference on Nanotechnology,* pp. 399-402, 2007.

[47] M. Zhao, K. Qiu, Y. Xie, J. Hu and C. J. Xue, "Redesigning software and systems for non-volatile processors on self-powered devices," *IFIP/IEEE International Conference on Very Large Scale Integration,* pp. 1-6, 2016.

[48] F. Su, Z. Wang, J. Li, M. F. Chang and Y. Liu, "Design of nonvolatile processors and applications," *IFIP/IEEE International Conference on Very Large Scale Integration,* pp. 1-6, 2016.

[49] X. Li, K. G. Ma, S., J. Sampson and V. Narayanan, "Enabling Internet-of-Things: Opportunities brought by emerging devices, circuits, and architectures," *IFIP/IEEE International Conference on Very Large Scale Integration,* pp. 1-6, 2016.

[50] T. Aoki, Y. Okamoto, T. Nakagawa, M. Ikeda, M. Kozuma, T. Osada and M. Fujita, "Normally-off computing with crystalline ingazno-based fpga," *IEEE International Solid-State Circuits Conference,* pp. 502-503, 2014.

[51] S. C. Bartling, S. Khanna, M. P. Clinton, S. R. Summerfelt, J. A. Rodriguez and H. P. McAdams, "An 8MHz 75µA/MHz zero-leakage non-volatile logic-based Cortex-M0 MCU SoC exhibiting 100% digital state retention at V DD= 0V with< 400ns wakeup and sleep transitions.," *IEEE International Solid-State Circuits Conference,* pp. 432-433, 2013.

[52] N. Sakimura, Y. Tsuji, R. Nebashi, H. Honjo, A. Morioka, K. Ishihara and T. Endoh, "A 90nm 20MHz fully nonvolatile microcontroller for standby-power-critical applications," *IEEE International Solid-State Circuits Conference ,* pp. 184-185, 2014.

[53] P. F. Chiu, M. F. Chang, C. W. Wu, C. H. Chuang, S. S. Sheu, Y. S. Chen and M. J. Tsai, "Low store energy, low vddmin, 8t2r nonvolatile latch and sram with vertical-stacked resistive memory (memristor) devices for low power mobile applications," *IEEE Journal of Solid-State Circuits,* vol. 47, no. 6, pp. 1483-1496, 2012.

[54] J. M. Brader, W. Senn and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural computation,* vol. 19, no. 11, pp. 2881-2912, 2007.

[55] S. Han, H. Mao and W. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR, abs/1510.00149,* vol. 2, 2015.

[56] E. Stromatias, D. P. M. Neil, F. Galluppi, S. B. Furber and S. C. Liu, "Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms," *Frontiers in neuroscience,* vol. 9, 2015.

[57] L. Muller and G. Indiveri, "Rounding methods for neural networks with low resolution synaptic weights," *arXiv preprint,* p. arXiv:1504.05767, 2015.

[58] M. Courbariaux, Y. Bengio and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015.

[59] F. Alibart, E. Zamanidoost and D. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nature communications,* vol. 4, 2013.

[60] Y. Pershin and M. D. Ventra, "Memory effects in complex materials and nanoscale systems," *Advances in Physics,* vol. 60, no. 2, p. 145–227, 2011.

[61] B. Linares-Barranco, et.al, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," *Frontiers in neuroscience,* vol. 5, no. 26, 2011.

[62] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience,* vol. 9, 2015.

[63] B. Nessler, M. Pfeiffer, L. Buesing and W. Maass, "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity," *PLoS Comput Biol,* vol. 9, no. 4, 2013.

[64] A. Serb, J. Bill, A. Khiat, R. Berdan, R. Legenstein and T. Prodromakis, "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses," *Nature Communications,* vol. 7, no. 12611, 2016.

[65] M. Graupner and N. Brunel, "Calcium-based plasticity model explains," *Proceedings of the National Academy of Sciences,* vol. 109, p. 3991–3996, 2012.

[66] C. Clopath and W. Gerstner, "Voltage and spike timing interact in STDP– a unified model," *Frontiers in Synaptic Neuroscience,* vol. 2, no. 25, p. 1–11, 2010.

[67] C. Mayr, P. Staerke, J. Partzsch, L. Cederstroem, R. Schüffny, Y. Shuai and H. Schmidt, "Waveform driven plasticity in BiFeO3 memristive devices: model and implementation," in *Advances in Neural Information Processing Systems*, 2012.

[68] H. Mostafa, A. Khiat, A. Serb, C. G. Mayr, G. Indiveri and T. Prodromakis, "Implementation of a spike-based perceptron learning rule using TiO2-x memristors," *Frontiers in neuroscience,* vol. 9, 2015.

[69] H. Mostafa, C. Mayr and G. Indiveri, "Beyond spike-timing dependent plasticity in memristor crossbar arrays," in *IEEE International Symposium on Circuits and Systems*, 2016.

[70] E. O. Neftci, B. U. Pedroni, S. Joshi, M. Al-Shedivat and G. Cauwenberghs, "Stochastic Synapses Enable Efficient Brain-Inspired Learning Machines," *Frontiers in Neuroscience,* vol. 10, p. 241, 2016.

[71] R. Naous, M. Al-Shevidat, E. Neftci, G. Cauwenberghs and K. Salama, "Memristive neural networks: Synaptic versus neuronal stochasticity," *AIP Advances,* vol. 6, no. 111304, 2016.

[72] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar and K. Bernstein, "Scaling, power, and the future of CMOS," in *IEEE International Electron Devices Meeting*, 2005.

[73] F. Pan, S. Gao, C. Chen, C. Song and F. Zeng, "Recent progress in resistive random access memories: materials, switching mechanisms, and performance," *Materials Science and Engineering: R: Reports,* vol. 83, pp. 1-59, 2014.

[74] A. Chen, "Area and thickness scaling of forming voltage of resistive switching memories," *IEEE Electron Device Letters,* vol. 35, no. 1, p. 57–59, 2014.

[75] B. Govoreanu, G. S. Kar, Y. Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini and N. Jossart, "10× 10nm 2 Hf/HfO x crossbar resistive RAM with excellent performance, reliability and low-energy operation," in *IEEE International Electron Devices Meeting*, 2011.

[76] H. S. P. Wong, H. Y. Lee, S. Yu, Y. S. Chen, Y. Wu, P. S. Chen and M. J. Tsai, "Metal–oxide RRAM," *Proceedings of the IEEE,* vol. 100, no. 6, pp. 1951-1970, 2012.

[77] S. B. Eryilmaz, D. Kuzum, R. Jeyasingh, S. Kim, M. BrightSky, C. Lam and H. S. P. Wong, "Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array," *arXiv preprint,* vol. arXiv:1406.4951, 2014.

[78] M. J. Lee, Y. Park, B. S. Kang, S. E. Ahn, C. Lee, K. Kim and Y. H. Kim, "2-stack 1D-1R cross-point structure with oxide diodes as switch elements for high density resistance RAM applications," in *IEEE International Electron Devices Meeting*, 2007.

[79] E. Linn, R. Rosezin, C. Kügeler and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature Materials,* vol. 9, no. 5, pp. 403-406, 2010.

[80] I. E. Ebong and P. Mazumder, "Self-controlled writing and erasing in a memristor crossbar memory," *IEEE Transactions on Nanotechnology,* vol. 10, no. 6, p. 1454–1463, 2011.

[81] G. W. Burr, et.al, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 Synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices,* vol. 62, no. 11, pp. 3498-3507, 2015.

[82] P. Y. Chen, et.al, "Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip," in *Design, Automation & Test in Europe*, 2015.

[83] J. Park, S. Ha, T. Yu, E. Neftci and G. Cauwenberghs, "A 65k-neuron 73-mevents/s 22-pj/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *IEEE Biomedical Circuits and Systems Conference*, 2014.

[84] J. Park, T. Yu, S. Joshi, C. Maier and G. Cauwenberghs, "Hierarchical Address Event Routing for Reconfigurable Large-Scale Neuromorphic Systems," *IEEE Transactions on Neural Networks and Learning Systems,* Vols. 1 - 15, no. 99, 2016.

[85] S. Joshi, S. Deiss, M. Arnold, J. Park, T. Yu and G. Cauwenberghs, "Scalable event routing in hierarchical neural array architecture with global synaptic connectivity," in *IEEE International Workshop on Cellular Nanoscale Networks and their Applications*, 2010.