Low-Power Computing with Neuromorphic Engineering

Dingbang Liu, Hao Yu, and Yang Chai**

The increasing power consumption in the existing computation architecture presents grand challenges for the performance and reliability of very-large-scale integrated circuits. Inspired by the characteristics of the human brain for processing complicated tasks with low power, neuromorphic computing is intensively investigated for decreasing power consumption and enriching computation functions. Hardware implementation of neuromorphic computing with emerging devices substantially reduces power consumption down to a few mW cm^{-2}, compared with the central processing unit based on conventional Si complementary metal–oxide semiconductor (CMOS) technologies (50–100 W cm^{-2}). Herein, a brief introduction on the characteristics of neuromorphic computing is provided. Then, emerging devices for low-power neuromorphic computing are overviewed, e.g., resistive random access memory with low power consumption ($<$ pJ) per synaptic event. A few computation models for artificial neural networks (NNs), including spiking neural network (SNN) and deep neural network (DNN), which boost power efficiency by simplifying the computing procedure and minimizing memory access are discussed. A few examples for system-level demonstration are described, such as mixed synchronous–asynchronous and reconfigurable convolution neuron network (CNN)–recurrent NN (RNN) for low-power computing.

1. Introduction

Massive data centers and cloud computing infrastructures enable abundant data exchange over the Internet. As the demand for data transfer has increased, the number of data-generating devices and their power consumption have grown exponentially in the past few years. According to recent reports, the overall energy usage of information infrastructure in 2018 was estimated as 198 terawatt hours or almost 1% of demand for electricity in the world.[1] This ratio will continue to increase substantially in the era of abundant data, resulting in an unmanageable level of power consumption.[1] The overall power consumption of a computing system is determined by the data processing procedure as well as the power consumption of each single device and their density. The development of low-power devices has been overviewed in existing literatures by adopting either new device structures or new materials.[2–6] It has become more important to develop highly efficient processing units to minimize the overall power consumption of computing systems.

Dennard predicted the reduction of switching power with the downward scaling of the feature size of the transistor,[7] and this trend ended around 2004. The operating voltage and current reduction of the transistor stagnated despite the ever-shrinking size of the transistor because of the increasing leakage current. The increase in operating frequency and the transistor density further exacerbates the power consumption of a computing system, which generates a large amount of heat and impedes power efficiency. Another challenge for power consumption is the intensive data transfer between data processing and memory units in conventional Von Neumann or Princeton computing architecture, the so-called “memory wall.”[8] In this computing paradigm, the computing unit can only process one task at a certain interval and wait for memory to update its results, because both data and instructions are stored in the same memory space, which greatly limits the throughput and causes idle power consumption. Although mechanisms like cache and branch prediction can partially eliminate the issues, the “memory wall” still poses a grand challenge for massive data interchanging in modern processor technology.

The prevalent synchronous clock design exacerbates the power problem, because the updates of all state-holding units are driven by a uniformly distributed global clock. Regardless of the usefulness of the task, all units are forced to respond to the arrival of clock edge. Thus, some unassigned units run idly and consume unnecessary power. In contrast to synchronous clock distribution, asynchronous distribution uses handshaking. The unparallelism issues also confine the power efficiency of high-performance computers. To fight against clock skew caused by uneven data transmission, a global clock with a high slew rate and elaborated frequency margin is required, which scarifies the area and power consumption.[9] As a result, a synchronous

D. Liu, Prof. H. Yu
School of Microeletronics (SME)
Southern University of Science and Technology
Shenzhen 518000, P. R. China
E-mail: yuh3@sustech.edu.cn

Prof. Y. Chai
Department of Applied Physics
Hong Kong Polytechnic University
The Kowloon, Hong Kong, P. R. China
E-mail: ychai@polyu.edu.hk

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202000150.

DOI: 10.1002/aisy.202000150

unparallel system will suffer a lot in power efficiency and latency.

To address the power problem of modern computing technology, researchers have attempted various computing paradigms, including parallel computing,[10,11] mixed synchronous–asynchronous,[9] analogue computing,[3,12] in-memory processing,[13,14] and neuromorphic computing.[9,15,16] In this article, we will focus on neuromorphic computing in the device, circuitry, algorithm, and system-level implementation. In section two, we will briefly introduce the basic concepts and characteristics of neuromorphic computing. In section three, we will describe the working principle of emerging devices for neural networks (NNs), mainly nonvolatile memory (NVM) devices, and different NN architectures. In section four, we will overview the characterized algorithm or treatment for each NN in the training procedure. In section five, we will provide a systematic overview of neuromorphic computing based system on chip (SoC), including TrueNorth, NeuroGrid, and Loihi.

2. Brief Introduction of Neuromorphic Computing

Neuromorphic engineering adopts computing architecture with highly interconnected elements (neurons), which features with analogue, parallel, and in-memory computing characteristics and provides opportunities for reducing power consumption of a computing system.[17] Compared with digital computing, analogue computing can deal with ambiguous and continuous inputs and allow real-time adaption in response to electrical stimuli. Because analogue computing usually requires less delicate external driving circuitries,[16,18] it has merits in terms of power consumption and fabrication cost.

Proposed by Flynn[11], parallel computing is also a way to achieve high power efficiency and can be divided into two sectors: data-level parallelism (DLP) that allows simultaneous multiple data processing and task-level parallelism (TLP) that enables independent operation of tasks. For instance, DLP can be achieved by using techniques like pipelining, out-of-order execution, and one-to-all parallel broadcasting in graphical processing units (GPUs), whereas TLP adopts multiple parallel threads and decoupled operating clusters.[10]

To break “memory wall,” in-memory processing has been studied since 2000s and regarded as a promising way to reduce redundant data movement between memory and processing unit and decrease power consumption. The concept has been implemented with different hardware, e.g., 3D-stack dynamic random access memory (DRAM)[19] and embedded FLASH.[20] Software solutions like pruning, quantization, and mixed precision topologies are implemented to reduce the intensity of signal interchanging.

The biological NN outperforms conventional Von Neumann computing architecture based on the central processing unit (CPU) or GPU in terms of power efficiency, which consumes only 1–100 fJ per synaptic event when performing intense parallel computing.[21,22] Currently, the synaptic resistive random access memory (RRAM) has been successfully demonstrated to exhibit low power consumption ($<$ pJ).[23] However, even the two most promising fully CMOS-based brain-inspired chips, Loihi from Intel and TrueNorth from IBM, still lag behind in power efficiency, which consume 23.6 pJ and 26 pJ per synaptic event.[15,24] Neuromorphic computing mimics synaptic dynamics in device and dataflow/connection in NNs to perform computing with low power consumption. The synaptic events, which are directly monitored and precisely adjusted by ion flux in memory devices, have been successfully emulated in the system based on RRAMs.[23] Neuromorphic computing architecture has a less complex peripheral driving circuit and simple weight modification procedure, e.g., fast dot product,[25,26] in comparison with conventional Von Neumann computing architecture.[27] Neuromorphic computing also emphasizes the imitation of systematic neural architecture, e.g., artificial NN (ANN),[28] convolution neuron network (CNN),[29] spiking NN (SNN),[15,30] and recurrent NN (RNN).[31] Unlike the general purposed processor, NNs are constructed for specified application scenarios. For examples, CNN has segregate layers and modularized kernels for fast and accurate feature recognition; the recycling output of RNNs makes it suitable for speed recognition; SNN is an event driving system and much more flexible and fault tolerant. The specialized neuromorphic computing architecture-based system on chip (SoC) usually exhibits several orders of magnitude reduction in power consumption when compared with the systems based on conventional Von Neumann architecture. For instance, the Intel produced Loihi chip,[15] which is based on asynchronous SNN and outperforms a single CPU by three orders of magnitude in both power efficiency and area.

Conventional computer systems based on Von Neumann architecture and Si CMOS technologies offer a much faster data processing speed and higher computing accuracy than the human brain but still significantly lag behind in terms of power consumption. For example, Google produced a supervised deep learning machine, Alpha Go, that consists of up to 1920 CPUs and 280 GPUs to defeat the world champion of the game Go, whereas it consumes 50 000 times more energy than the human brain.[32] Recent advances on the algorithms of neuromorphic computing, involving effectively solving complex tasks like the human brain, such as, pattern recognition, data classification, and object detection, make it close to practical applications. However, the hardware implementation of these neuromorphic computing algorithms based on Si CMOS technologies for parallelism and weight adaption poses great challenges in terms of computation, memory, and communications and gives rise to high power consumption. It is highly imperative for a breakthrough in hardware implementation for neuromorphic computing systems.

3. Emerging Devices for Neuromorphic Computing

As today’s computing becomes more data intensive, the computing paradigm is transferred from computation centric to data centric. Neuromorphic computation (e.g., vector matrix multiplication, etc.) requires device characteristics with linear and symmetric tuning, long retention, low stochastic behavior (blind updates), and low energy tuning. Emerging nonvolatile memories (NVMs) can potentially meet these requirements, e.g., instantaneous writing/reading ($\approx$ns), low operation voltage ($<$1 V), and power consumption ($<$ pJ per synaptic event).[33]

Here, we briefly describe the working mechanisms of RRAM, spin-transfer-torque-magnetic random access memory (STT-MRAM), phase change memory (PCM), conductive bridge random access memory (CBRAM), and optoelectronic RRAM (ORRAM).[34–36]

The widely used static/dynamic random access memories (SRAMs and DRAMs) store information through charge and require a constant wave of operating voltage to maintain or update their states, by charging or discharging a capacitor. RRAM is noncharge-based device, which is free from the constraint of a capacitor for data storage. A small operating voltage (<1 V) is sufficiently large for operation RRAM, and no subsequent power is required for maintaining the states.

Today's deep learning applications require unprecedented amount of data for dynamic data analysis. Conventional CPU/GPU has to fetch data from cache memory through programmed procedures, which usually result in the standby of processors due to lack of useful data. Because data interchanging is costly in latency and power consumption, the most direct optimization methodology is data localization, which minimizes the physical distance between the processor and cache. RRAM has good compatibility with CMOS, which makes it ideal for 3D stacking. The 3D-stacked RRAMs can be mounted on the top of the processor. Its thin layout cannot only shorten the data travelling path, but also enable high-density memory. The 3D-stacked RRAM memory can substantially reduce data travelling, enlarge data reuse, and ultimately improve power efficiency. The emerging nonvolatile devices like RRAM, STT-MRAM, and PCM outperform state-of-the-art memories, such as DRAM and FLASH in terms of cell size, power consumption, and read/write speed. In STT-MRAM, retention time is in trade-off with the operating or switching power. Unlike other NVM devices with retention time over 10 years, the retention of STT-MRAM has a one-to-one relation with stimulating pulse.[6]

As a memory, reading/writing is essential for computing. Unlike the traditional state-holding units, RRAM can be read out and modulated in a crossbar separately with high power efficiency. However, during the reading/writing process, only the targeted unit, the one-half or one-third biasing on one-transistor-one RRAM (1T1R) crossbar, for instance, is assigned and all other units would remain idle.

3.1. Oxide Random Access Memory (OxRAM)

The resistance switching property of oxide random access memory (OxRAM) was first revealed in the 1960s.[37] The device configuration of OxRAM has a structure with an insulating material between two metal electrodes. It includes the top electrode (TE), the middle layer of oxides, and the bottom electrode (BE). Generally, the OxRAM has two-terminal devices, which have two resistance states, a high resistance state (HRS or OFF state) and a low resistance state (LRS or ON state). The resistance states are manipulated by applied electrical potential and stimuli duration. The resistive switching mechanism is a result of the formation/rupture of the conductive region in the metal oxide caused by the generation and drift of oxygen vacancies. It exhibits a symmetric behavior or Lissajous I/V curve in the SET/RESET process.[38] The noncharge-based RRAM possesses a long-term plasticity characteristic, which keeps the weight (conductance) stable for a certain period of time, even without cycling operating power.

In combination with all of the desirable properties, NVMs are inherently suitable for synaptic simulation.[39–41] We will describe the computing mechanism of NVM, as well as their merits of low-power computing. The resistive switching property of OxRAM is closely related to the migration of O^{2-} vacancy in the oxide layer, as shown in **Figure 1**. The gap distance (g) from the TE to an O^{2-} vacancy-rich margin is a determinant factor in the conductance adjusting mechanism. E_{ag} and E_{ar} stand for the activation energy of electron generation and recombination process. Generally, E_{ag} is larger in orders of magnitude than E_{ar} for a gradually shifting phase from the LRS to HRS. In the SET procedure, oxygen ions are driven by the applied electrical field and overcome E_{ag} to migrate toward the TE and leave oxygen vacancies in the oxide, resulting in a LRS. During the RESET process, a reverse electrical field is applied and forces the oxygen ions to return and recombine with oxygen vacancies, giving rise to a HRS. The I–V characteristic of OxRAM is shown according to Equation (1)

$$I = I_0 \exp\left(-\frac{g}{g_0}\right) \sin h\left(\frac{V}{V_0}\right) \tag{1}$$

where $I_0(\approx 1\text{ mA})$, $g_0(\approx 0.25\text{ nm})$, and $V_0(\approx 0.25\text{ V})$ are constants. The HRS exhibits an exponential corelationship with distance (g) and a large amount of V (>0.5 V). The OxRAM resistance responds not only to the magnitude of electrical stimuli, but also depends on the duration of applied stimuli, as shown in Equation (2).

$$\begin{aligned}\frac{d_g}{d_t} = -V_0\Bigg[&\exp\left(-\frac{qE_{ag}}{kT}\right)\exp\left(\frac{\gamma a_0}{L}\frac{qV}{kT}\right)\\ &-\exp\left(-\frac{qE_{ar}}{kT}\right)\exp\left(-\frac{\gamma a_0}{L}\frac{qV}{kT}\right)\Bigg]\end{aligned} \tag{2}$$

where $\frac{d_g}{d_t}$ is vacancy-generating speed and determined by the discrepancy between the generation and recombination rate. The mathematic model of OxRAM is similar to the biological synapse and non-volatile behavior with ultralow (sub-pJ) power computing is exhibited,[42–44] which makes OxRAM an ideal

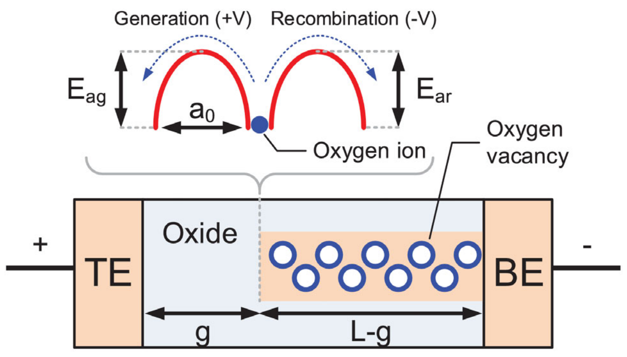


Figure 1. Tunneling mechanism of RRAM. Form and disintegration of conductive region by means of oxygen-ion status transitions. Reproduced with permission.[33] Copyright 2016, Morgan & Claypool Publishers.

candidate for the building block of neuromorphic computing. Spiking time-dependent plasticity, a vital characteristic of the biological synapse for array-level neuromorphic computing, e.g., SNN, has been successfully demonstrated in OxRAM.[45]

As a two-terminal nonvolatile device, the OxRAM crossbar has inherent advantages in power efficiency and switching speed.[46,47] When OxRAM crossbar are fabricated in a large scale with CMOS integration, OxRAM devices are sandwiched by two perpendicular electrode wires at every cross point. Thus, a matrix-like dense array of nanoscale OxRAM is relatively easy to implement. As nonvolatile and two-terminal OxRAM enables in-memory computing and requires only a small amount of operating voltage, the processing unit based on the OxRAM crossbar[48] outperforms CPU or GPU with conventional Von Neumann architecture in energy consumption, scalability, and process speed.

3.2. Spin-Transfer-Torque Magnetic Random Access Memory

The STT-MRAM contains three layers to form a magnetic tunneling junction (MTJ), the free layer (FL) for data storage, the pinned layer (PL) for reference, and a tunnel barrier for magnetic insulation and data reading procedure.[49–52] The tunnel barrier defines the energy barrier (E_B) for switching ON and OFF states. The magnetic field direction of the FL is deterministic to the electrical resistance of the memory cell. When the magnetic moments of the FL and tunnel layer are in parallel, the STT-MRAM shows low resistance. STT-MRAM exhibits a HRS in antiparallel configuration. By introducing an opposite magnetic field into the FL, STT-MRAM can exhibit multiple resistance states.[50,53] An insulation domain wall is placed between the two opposite magnetic domains and adjusted by the STT phenomenon. The STT phenomenon has been successfully demonstrated on MgO-based MTJ,[54] which enables the alternation of the direction of the magnetic field in the FL, when applying operating current with a certain direction to the target STT-MRAM cell. As the retention time is associated with E_B and switching power, STT-MRAM shows the smallest power consumption and fastest operating speed in all NVMs.

3.3. Phase Change Memory

In PCM, the transition from the HRS to LRS is represented by the phase change from the amorphous to the crystalline state of the switching layer. In the RESET procedure, the crystallized phase change material is melted to the amorphous phase by electrically generated heat. A smaller current is required to set the phase change material back to the LRS.

Compared with other NVM devices, PCM has a relatively large range of switching resistance, which makes it ideal for synaptic analog computing with a modulation scale of 1% per event. Using a gradually increasing amplitude and customized duration of the stimulating impulse, researchers can tune the resistance of the PCM with an order of magnitude change through 100 steps from the LRS to HRS.[55] As shown in **Figure 2**, plasticity or weight updating of PCM is time dependent. The different arrivals of the pre and postsynapse event determine the direction of modulation. Like neurosynapse, PCM exhibits accumulative behaviors in response to a string of continuous spikes and can be described according to Equation (3)

$$\Delta w = Ae^{-\Delta t/\tau} \tag{3}$$

where Δw is the synaptic weight to be updated, Δt is the arrival time variance of pre-/postsynapse events, A is the scaling factor, and τ stands for the constants of the spike time dependent plasticity (STDP) curve. The PCM-based synapse exhibits lower power consumption compared with CMOS (SRAM) and a capacitor-based synaptic device (DRAM), which consumes only ≈50 pJ and 0. 675 pJ for the RESET and SET process and can be further reduced as the PCM device shrinks down.[55] But the CMOS and capacitor-based synaptic device require consumption of nJ per synaptic event for proper functionality.[56]

The NVM devices exhibit great advantages in terms of endurance, power consumption, and speed over SRAM and FLASH in data storage. A RRAM-based hybrid system can work as a memory and processor simultaneously, which can eliminate redundant data movement. With the boost of the dot-product engine,[25,26] an array of RRAM can solve some complex tasks

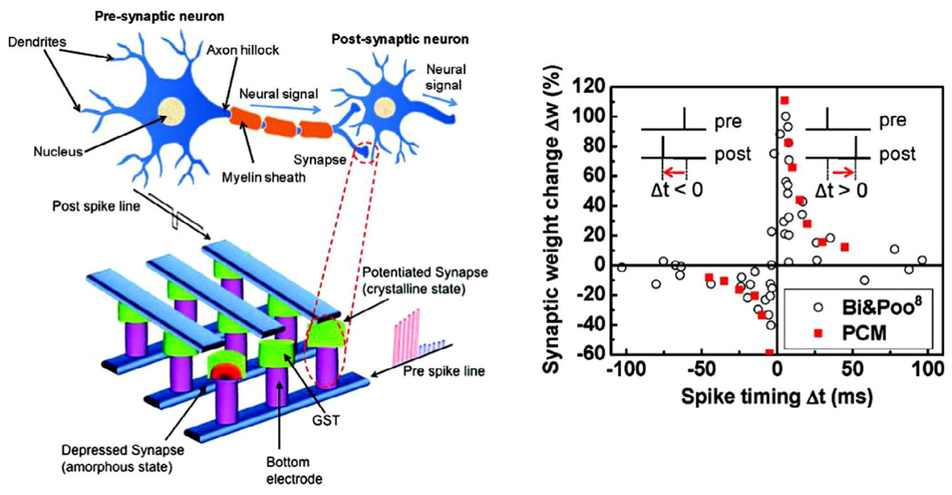


Figure 2. Simulation of PCM to neural cell, and spiking time-dependent plasticity of the biological synapse and PCM. Reproduced with permission.[55] Copyright 2011, American Chemical Society.

by simply using Ohm's law and Kirchhoff's current law in a very short time, e.g., the multiplication and summation of matrix.

3.4. Conductive Bridge Random Access Memory

CBRAMs inherit the electro-chemical resistive switching mechanism of RRAM,[57] which is determined by the forming/dissipation process of conductive region. CBRAM and OxRAM share a lot of similarities in functionalities. There are some differences between them in device structures. The metal oxide in OxRAM is replaced with solid electrolyte in CBRAM. To generate metal ions, an oxdizable metal is used as the top (active) electrode.[53,58]

3.5. Characteristics of NVM Device as Synapse for Neuromorphic Computing

The desirable characteristics of the neuromorphic device for computing include linearity, symmetrical tuning, scalability, multilevel resistance state, long retention time, stochastic behavior, and low energy consumption.

1) Linear/symmetric tuning: Because weight updating follows a time-dependent mechanism, the linear and symmetric relationship between the weight carrier and stimulating pulse facilitates a less computing operation and decrease in power consumption. It requires NVMs to possess relatively linear I/V electrical characteristics. For instance, the linearity of TaO_x/TiO_2 RRAM can be enhanced by applying a string of varying-duration impulses, as shown in **Figure 3**. We also desire NVMs to possess symmetric characteristics, because the implementation of NN requires the same amount of resistance change in both the potentiation and depression process.

2) Scalability and stackability: The downward scaling of the memory device enables high-density integration. Most of the NVM devices have been successfully fabricated at an $\approx$10 nm scale level.[59] To further increase the integration density and shorten the distance between memory and process, emerging NVM devices are implemented in 3D stacking and embedded into the processor.[48] To enable 3D integration, it requires the process temperature of NVMs to be as low as possible.

3) Multilevel resistance state: As the weights in biological synapses are processed in the analog form, it requires multilevel resistance states in neuromorphic devices. A great amount of bioinspired algorithms requires multilevel states than the inferenced 0/1 state.[60] The ideal bioinspired synaptic device has over 100 states of resistance level and shows a higher performance and variance robustness than those with less resistance states.

4) Long-time retention: During the implementation process of computation, the continuous updating of weights has limited requirements on retention time. The traditional SRAM/DRAM array utilizes capacitors to maintain the electrical potential during the refresh interval of the system, which has a general standard of 64 ms.[61] But, after the operation, the long retention time of NVMs is required to ensure that the stored weight remains even if NVM devices are switched to idle (power off). Therefore, extra power can be saved because a global updating of weights is no longer required in the NVM-based system.

5) Low stochastic: NVMs still suffer from device variation and switching ambiguity, e.g., abrupt decrease in resistance in OxRAM and PCM. The stochastic switching property of NVM is still open research and hinders commercial applications. Recently, there have been a few works on NNs,[60,62] which use a novel weight-updating strategy to compensate the accuracy loss caused by the nonlinearity of RRAM device.

4. NN Models

To achieve state-of-the-art deep learning algorithms, researchers tend to aggressively extend the depth and frame of NNs, when dealing with enormous amount of datasets and ambiguous patterns. For example, the ResNet by Microsoft has a number of layers up to 1000 on CIFAR-10 dataset with a fairly good training accuracy of 92.07% .[63] With frequent memory data fetching on traditional Von Neumann processors, e.g., GPU/CPU, the power efficiency and latency flaws are becoming more obvious. In this

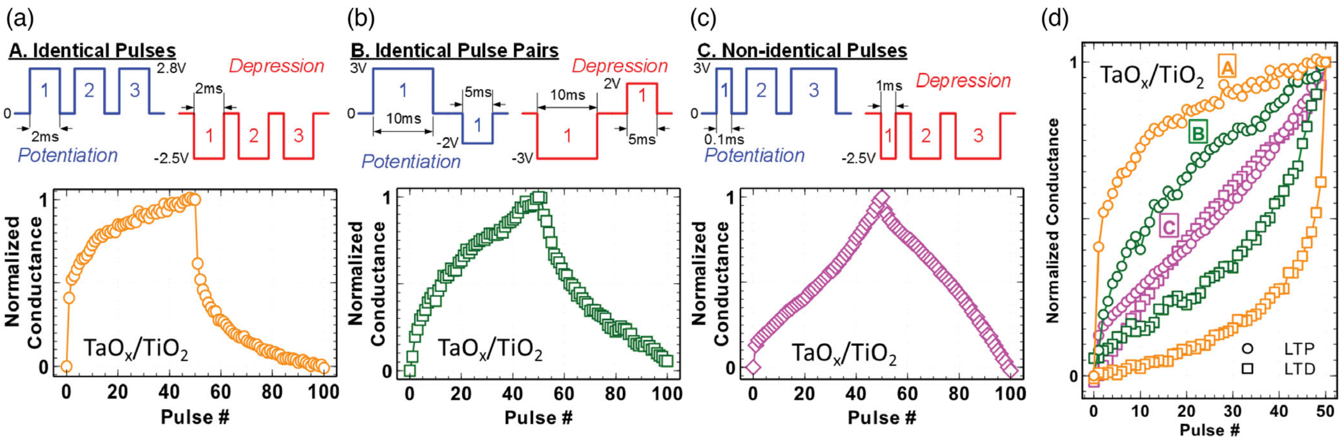


Figure 3. Modulation of TaO_x/TiO_2 RRAM stimulating pulse for better linearity. Date from reference. Methodology A is identical simple spike trains for both potentiation and depression which lead to the largest nonlinearity. Methodology B uses both negative and positive spikes for potentiation and depression, where the negative spikes cancel the overshoot of positve spikes to increase linearity. Methodology C has adjustable spike duration with respect to the resistance state of NVM. The preliminary ideal of such a nonlinearity migitation approach is to slow down the most drastic switching process (the beginning of LTP and LTD). Reproduced with permission.[27] Copyright 2018, IEEE.

section, we will discuss the implementation of NNs with NVMs as well as the training algorithms.

4.1. Spiking NN

Inspired by the biological information movement that uses a constant stream of electrical pulses to communicate and compute, spiking NN (SNN) adopts spikes (in the analog system) or squares (in some digital systems) to represent the arrival of events as a medium of data transferring. The rate-driven NNs[64] (e.g., traditional DNN) use continuous signals (e.g., optical flow in static images and vibration of sound) for training, which is measured based on differentiation operation. Instead of being driven by rate, SNN is an event-driven system, which transforms the continuous fluctuation of the dataset, e.g., the value of pixels and intensity of acoustic feature, into spike trains. SNN can scale the membrane potential of each synapse to represent the excitatory and inhibitory behavior and utilize sparsely distributed, highly informative spikes to reduce power consumption during information processing. In the SNN, only the synapses driven by the event respond to the arrival of spikes, and the rest of the synapses remain idle during the implementation of computing. Therefore, SNN is potentially more energy efficient than traditional DNN. The SNN emphasizes the accumulative behavior of the synaptic weight instead of a continuous modulation, where the weights can remain constant unless the accumulated features reach the critical point (threshold). In general, the firing frequency of SNN is determined by the intensity of the incoming spikes. The stronger the input spikes arrive, the earlier the output spike fires.

SNN is a computing model that more closely mimics natural NNs. It not only uses sparse spike trains to communicate but also localizes the memories adjacent to the processing element (PE) and further minimizes data movement. For instance, the relatively matured SNN neuromorphic chips, TrueNorth and Loihi, consume only 45 pJ and 24 pJ per synaptic event, respectively. Caused by their new methods for coding information, the SNN with conventional CMOS still suffers from a large size, low parallelism, and high power consumption as SRAM is largely used in weight storage, e.g., each neuron in combination with 1 kB SRAM in TrueNorth. The sequential updating of SRAM not only increases system delay but also deteriorates bit accuracy and occupies a great amount of on-chip area. The large power consumption in the reading/writing process of SRAM also restricts the applications of novel neuromorphic chips.[65]

The first SNN model was proposed by Hodgkin and Huxley.[45,66] Due to the high cost of the Hodgkin and Huxley SNN model in computation, more simplified models were proposed, e.g., the Izhikevich model[67] and the leaky integrated-and-fire (LIF) model.[68] The intuitive LIF model gains popularity because of its direct description of spike firing in response to accumulating stimuli. The most matured neuromorphic chips TrueNorth[9] and Loihi[15] also adopted the LIF model as the fundamental neuron block. In the LIF model, the transmission of spikes starts transmission from preneuron to postneuron via synapses and accumulates (dot product) in the receiving side as the postsynaptic potential, as shown in **Figure 4**. Because of the sparsely distributed spikes, SNN is intrinsically propelled by derivative-based measures, such as back propagation. Thus, most pristine SNNs are constructed in the single layer, and deep SNN is usually constructed in combination with other types of NNs (ANN and CNN) for compatibility with differentiable approaches and computing power efficiency. To have a better illustration of SNN, we use a simplified model to show the fundamentally intrinsic architecture of SNN. A three-layer SNN (similar to ANN) with full connection and driven by spikes trains is shown in Figure 4a.

In general, the LIF neuron model contains three procedures: 1) synaptic integration, 2) leak integration, and 3) spiking fire and reset.[69] The accumulative behavior of postsynaptic can be described according to Equation (4)

$$V_t = V_{t-1} + \sum_{i=0}^{N} w_i x_i(t) \tag{4}$$

where V_t is the membrane potential at time t, V_{t-1} is the membrane potential of the previous spike, w_i is the weight of the i-th synapse, and $x_i(t)$ is the input feature map at time t. The updating of weight in SNN reveals a similar static–dynamic adaptation mechanism to the biological brain, such as

$$x_i(\mathrm{t}) = \begin{cases} 1, \text{if } t = t_{i,f} \\ 0, \text{otherwise} \end{cases} \tag{5}$$

The static weights stand for the intensity of previous synaptic events, and the dynamic weights indicate the fluctuation of weights in response to impending spikes.[70,71] Here, $t_{i,f}$ is the time when the i-th neuron fires. The leak integration can be described as

$$V_t = V_t - \lambda_i \tag{6}$$

where λ_i is the linear constant leakage of membrane potential and usually functions as a bias in time t. The firing and reset procedure of LIF can be described as

$$V_\mathrm{mem} = \begin{cases} V_\mathrm{spike}, V_\mathrm{mem} \geq V_\mathrm{threshold} \\ V_\mathrm{reset}, t \in (t_\mathrm{spike}, t_\mathrm{spike} + t_\mathrm{refractory}) \end{cases} \tag{7}$$

where $V_\mathrm{threshold}$ is the threshold voltage of the membrane potential, V_reset is the reset voltage after firing, and t_spike is the moment when neuron fires. The accumulative behavior of postsynapse is shown in Figure 4b and Equation (8).

$$\tau \frac{\mathrm{d}V_\mathrm{mem}}{\mathrm{d}t} = -V_\mathrm{mem} + \sum_i V_i w_i \tag{8}$$

τ is the leakage time constant without the presence of spikes. In the refractory interval (reset model), all synaptic movements are inhibited. As the receiving side of SNN, V_mem accumulates forthcoming spike trains with respect to synaptic weights and leak current with time constant τ. t_post is the moment when V_mem exceeds $V_\mathrm{threshold}$ and fire output spike.

STDP is the most prevailing training methodology in SNN, which modulates synaptic weights by exploiting the relative time difference of pre/postsynaptic events. The excitation/increase in weights is achieved by long-term potentiation (LTP), i.e., the

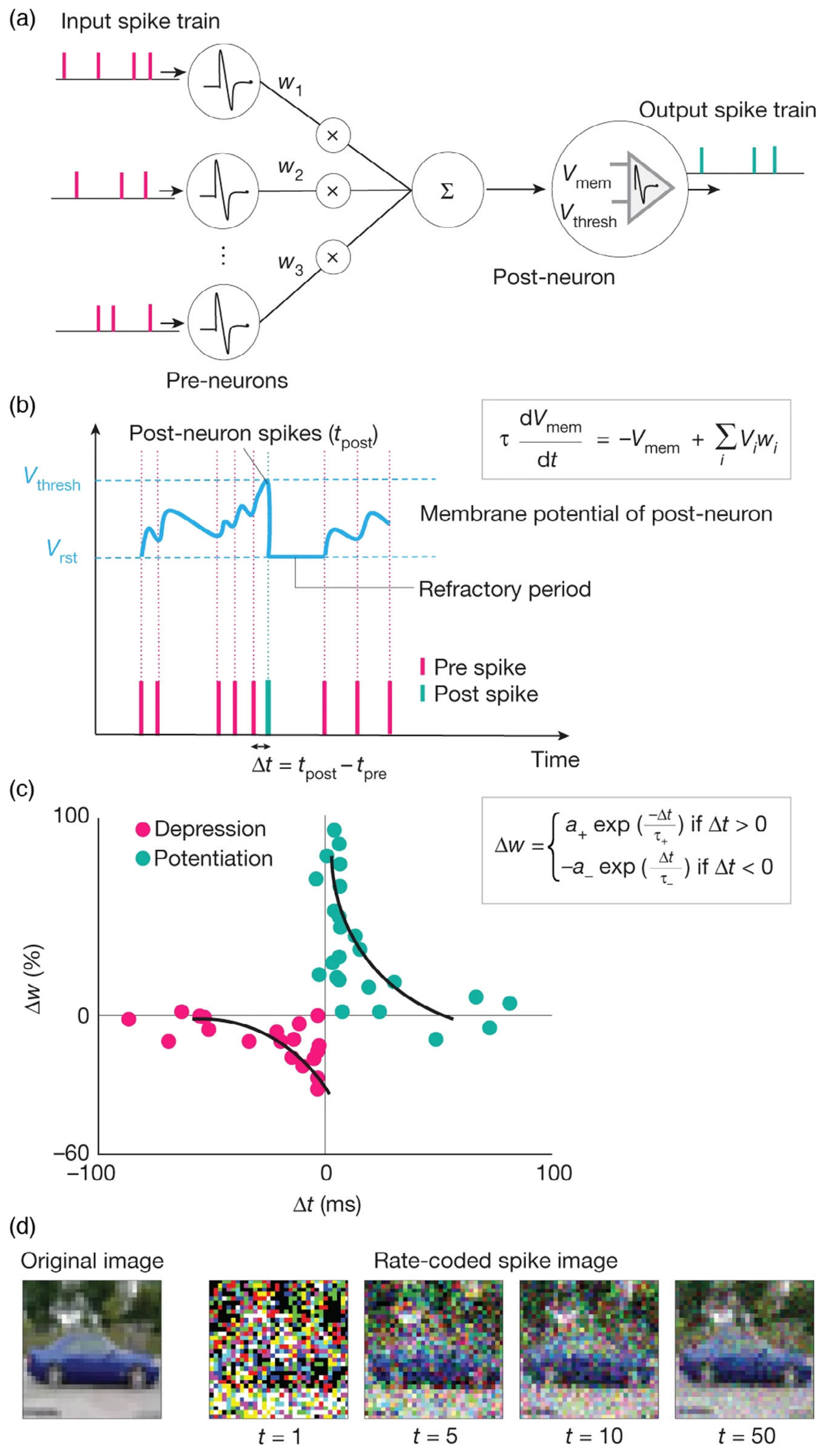

(a)
Input spike train
w_1
w_2
w_3
Σ
Output spike train
V_{mem}
V_{thresh}
Post-neuron
Pre-neurons
(b)
Post-neuron spikes (t_{post})
$\tau \frac{dV_{mem}}{dt} = -V_{mem} + \sum_i V_i w_i$
V_{thresh}
V_{rst}
Membrane potential of post-neuron
Refractory period
Pre spike
Post spike
Time
$\Delta t = t_{post} - t_{pre}$
(c)
Depression
Potentiation
$\Delta w = \begin{cases} a_+ \exp\left(\frac{-\Delta t}{\tau_+}\right) \text{ if } \Delta t > 0 \\ -a_- \exp\left(\frac{\Delta t}{\tau_-}\right) \text{ if } \Delta t < 0 \end{cases}$
Δw (%)
100
0
−60
−100
0
100
Δt (ms)
(d)
Original image
Rate-coded spike image
t = 1
t = 5
t = 10
t = 50

Figure 4. a) Spike transmission among synapses. Input spikes multiplied by synaptic weight w and summed on postneuron causing the fluctuation of membrane potential. Once V_{mem} exceeds V_{thresh}, the postneuron fires spike.b) Accumulative behavior of the single spike neuron network. c) Spike time-dependent plasticity of the SNN with long-term potential (LTP) and LTD in response to pre/postsynaptic time. d) Conversion of the conventional benchmark dataset (static image) to spike trains. Reproduced with permission.[30] Copyright 2019, Springer Nature.

occurrence of presynaptic events is earlier than postsynaptic events. The inhibition/decrease in weight is accomplished by long-term depression (LTD), for instance, early occurrence of postsynaptic events before presynaptic events. The STDP exhibits a temporal-based relationship between input spikes and output spikes, as shown in Figure 4c and Equation (9).

$$\Delta w = \begin{cases} a_+ e^{\frac{-|t_{\text{pre}} - t_{\text{post}}|}{\tau}}, t_{\text{pre}} - t_{\text{post}} \langle 0, a_+ \rangle 0 \\ a_- e^{\frac{-|t_{\text{pre}} - t_{\text{post}}|}{\tau}}, t_{\text{pre}} - t_{\text{post}} \geq 0, a_- < 0 \end{cases} \tag{9}$$

where Δw is updating weight, a_+ and a_- are constant for the learning rate of pre- and postsynaptic events, respectively, and τ is time constant for temporal scaling. The conversion from the static frame dataset, such as ImageNet to temporal based dataset, uses weight (temporal) rescaling and normalization to achieve the functionalities of SNN. The synaptic weights in STDP are inherently positive, whereas the nonlinear activation function, such as *tan*h and softmax, is adaptive to both positive and negative values. Therefore, the negative synaptic weights in SNN are either discarded or modulated to the positive domain by LTP. Such approaches of desertion and manipulation of weights are likely to induce accuracy loss in computing. Despite all of these aforementioned drawbacks, SNN with the converted dataset is still the most accurate spiking neuron model in the recognition process.

STDP enables SNN with great performance in power efficiency and computation speed, due to its spatio–temporal event-driven processing of information. Using sparse trains of spikes as the mean of information media, SNN not only reduces power consumption (idle power), but also ensures the accuracy of training in data transmission by means of STDP and/or other temporal coding. State-of-the-art DNN can satisfy the requirements of machine learning in accuracy, but still penalize in power consumption, especially in memory accessing. Temporal coding of SNN converts continuous flow of input into discrete spike trains,[30] which greatly reduce the complexity in computing and the interchanging of data between processor and memory. SNN is potentially capable of extending the application scenario for real-time sensor and training. SNN based on RRAM can potentially blur the boundary between memory and PE and shows low leakage current and high parallelism in row-by-row operation.[27,72] Currently, the long historical ANN or DNN is still dominant with their gradient-base backpropagation training methodologies, as well as their static image training dataset, e.g., ImageNet[73] and CIFAR.[74] SNN still lacks specified event-based multilayer training methodologies and train dataset. Thus, SNN training has to be evaluated in traditional neuron network benchmarks. Figure 4d shows the conversion of the static image into the temporal coded spike. Each pixel in the original image is transformed into Poisson spike trains with the spike firing rate related to pixel amplitude, and different time spectra represent the summation of rate-coded maps.[75] As shown in Figure 4d, the conversion of data imposes great latency issues to SNN training and impedes power efficiency improvement.[30]

4.2. SNN Training Topologies

The learning methodologies of SNN can be divided into two sectors: supervise learning and unsupervised learning. Supervised learning uses labeled features for training. The earlier supervised training algorithm, the Resume,[76] implements single-layer SNN training by exploiting STDP temporal coding for classification. Later, machine learning requires deeper layers for training accuracy. The training algorithms for SNN, e.g., SpikeProp,[77] tend to combine STDP with supervised gradient-based methodologies, such as backpropagation. One of the SpikeProp's distinctive characteristics is that the requirement for differentiable hidden units to feature maps is waived by backpropagating errors, and a constant spike train is used on the output feature map.[67] Multilayer SNN with error backpropagation has shown state-of-the-art accuracy in pattern recognition,[78–81] where backpropagation[65] can be described according to Equation (10)

$$\delta_j^\mu = g'(a_j^\mu) \sum_k w_{kj} \delta_k^\mu \tag{10}$$

where j and k stand for the indices of SNN units, the unit of j connects feed-forward to unit k (feedback from k to j), δ_j^μ and δ_k^μ represent the partial derivative between cost function δ^μ and random j and k, and g' is the activation function of unit j's input a_j^μ. However, backpropagation requires continuous g' for weight error calculation. The activation function of SNN is nondifferentiable. As a result, the backpropagation SNN replaces the nonderivable LIF model with a differentiable approximation,[82] though it is implausible in the biological term. To guarantee the functionality of backpropagation in SNN, the feed-forward weight w_{kj} is used in the feedback fashion, where the flow of feed-forward weights and feedback weights are absolutely symmetrical. Generally, symmetric feedback training topology reveals the best flexibility in solving various complex problems, whereas other feedback topologies can only process relatively simple tasks.[83]

Unsupervised learning helps to fully exploit the power advantage of SNN by localized training and STDP-based learning rules,[84,85] though it lacks in accuracy especially in the multilayer scenario. The merits of STDP of the localized spatio–temporal weights and sensitivity to repetitive spike trains in stochastic spike trains make it ideal for energy-efficient unsupervised learning, even with implicit time inference[86,87] when the training background is infused by noise. It has been proved that a full unsupervised SNN can achieve competitive training accuracy.[85]

4.3. Convolutional NN

A convolutional NN was first brought out by Lecun[88] and accomplished based on conventional Si CMOS technologies[89] and emerging memory devices.[90] Unlike the fully connected (FC) NN, CNN shares its weights to reduce training complexity and uses a small frame of input to strengthen the spatial relationship in the initial image. To satisfy the requirements in training accuracy, a power-hungry CPU/GPU is inevitable for CNN implementation and thus restrains wide applications of CNN. In addition, a conventional CNN based on Von Neumann architecture requires an intense interchange of information between the memory and processor, which further impedes the transplantation of CNN onto the portable device or edge computing.

A typical CNN is constructed by two fundamental elements, the feature extraction and the classification.[91]The feature extraction contains a convolution layer and pooling layer, and the classification is composed of an ANN-like FC feed-forward NN. During the convolution operation, different features are extracted by multiplexing and adding (dot-product) different convolution kernels and then generate the output feature map after nonlinear transformation. The convolutional operation is described as Equation (11)[92]

$$g(x,y,z) = \sum_{i=0}^{c-1}\sum_{j=0}^{c-1}\sum_{k=1}^{l} f(x+i,y+j,k)\cdot c_z(i,j,k) = \vec{f}\cdot\vec{c_z} \tag{11}$$

where $\vec{f}$ and g stand for the 3D input and output matrix, respectively, and $\vec{c_z}$ (i, j, k) represents the kernel matrix, which is usually smaller than the input matrix. To accelerate the convergence, a nonlinear function is attached to the convolution kernel, e.g., $\mathrm{ReLU}(g) = \max(g,0)$.

The shirk in computing size,[93] likewise 16- or 32-bit floating point to binary or ternary, enables the improvement in power efficiency and latency. Because weights and feature maps are stored in the single-bit format, the requirement for logic and resource is greatly reduced. Inspired by such topology, a topology named bitwise CNN (BCCNN)[94] is adopted to reduce the complexity of CNN, as described in Equation (12)

$$s_k = \sum_{i=1}^{N} w_k^b \otimes a_{k-1}^b = \sum_{i=1}^{N}\left(w_k^b\cdot a_{k-1}^b + \overline{w_k^b}\cdot\overline{a_{k-1}^b}\right) \tag{12}$$

where $w_k^b \in (+1, -1)$ and $a_{k-1}^b \in (+1, -1)$ stand for the kth stored binary weight and input feature map, respectively, and complement for computation in negative domain. Concerning the trade-off between accuracy and power efficiency, we present the quantized four-bit CNN for relatively high accuracy and low power,[95] with approximately four times improvement in power efficiency than four-bit ResNet-50 based on GPU implementation. The pooling layer is executed after nonlinear transformation to minimize the amount of parameters and local invariance, e.g., maximization or average. Computational complexity can be reduced by such measure.

In the fully constructed CNN, the convolution layer and FC layer have different specifications in data processing. Similarly, the convolution layer is mostly computing centric, and the FC layer is memory centric.[92] Compared with convolution layer, which consists of a great amount of operations (dot production), the FC layer has much more frequent access to memory. To address such challenges, RRAM is exploited in the CNN to enable fast "matrix vector multiplication" or dot production[25,96] with high power efficiency and hasten memory access by localization of memory.

Apart from the aforementioned feed-forward propagation, CNN uses backpropagation to update weights until error converge. The prevailing backpropagation was initially developed from the Euler–LaGrange formula to find the flattened part (solution) by iteration of chain rules and feeding error derivatives backward through Jacobian matrix.[97,98] The error backpropagation was first introduced into the NN by Werbos.[99]

4.4. CNN Circuits

The hardware-based accelerators are largely deployed in machine learning-related fields to enhance computing performance, e.g., improvement of power efficiency and throughput. However, the hardware-based accelerator implemented in traditional Von Neumann architecture still suffers from memory wall and results in severe degradation in terms of power efficiency and latency of the system. To alleviate the problems caused by Von Neumann architecture, we represent in-memory computing with NVMs.[6] The NVMs are capable of holding logic weights (1 or 0 or multistate) without a constant maintaining power. With Ohm's law and Kirchhoff current's law, the NVM array can perform fast convolution within a one-step procedure.[25] The long retention time of NVMs enables in-memory computing and avoids the frequent fetching of memory. Thus, the convolution procedure, generally the most time- and power-consuming procedure in machine learning, can be sped up by introducing the NVM array into vector-to-matrix multiplication. Here we present several NVM-based neuron networks as well as their merits in machine learning.

In a CNN based on memristor crossbar, kernels are represented by conductance or resistance of crossbars. The convolutional operation in the memristor crossbar is mainly the dot production of kernel matrix. At the very end of each column, a sense amplifier circuit is used for activation, e.g., sigmoid and scaling of output, as shown in **Figure 5**.

RRAM crossbar and NOR FLASH array are utilized in convolution for their linear adjustable conductance and long retention time. In NOR FLASH array, digital multiplication of the weight and feature map has been successfully demonstrated.[101–104] The multiplier stands for the input feature map fed from the world line (WL) on the gate side of FLASH, the multiplicand represents the threshold voltage V_{th} of the FLASH cell, and the results are expressed in the drain-to-source current I_{ds} and flow out through bit line (BL). The current on BL will be summed up by Kirchhoff current's law. Because V_{th} is preprogrammed into NOR FLASH array before computation and, the input feature map is fed simultaneously by the WL, and the convolution procedure can be accomplished within one step. However, currently, the NOR FLASH-based convolution accelerator still suffers from the endurance constraint of NOR FLASH. Generally, a NOR

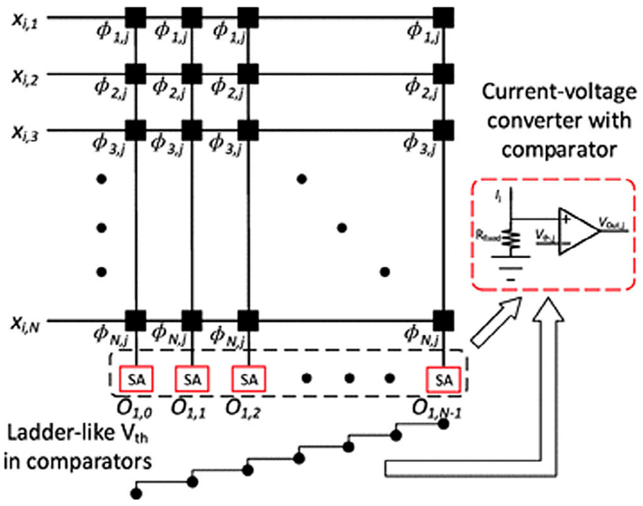


Figure 5. Multiply accumulation procedure in RRAM crossbar with a sensing amplifier above each column subarray for activation. Reproduced with permission.[100] Copyright 2017, Springer International Publishing AG.

FLASH device has a limited endurance of 10^5 times,[105] which is insufficient for the deep learning training process of modern standards and thus can only be used for inferring. Although there are methodologies[106] that can precisely tune NOR FLASH cells with desired train weights, the RRAM-based convolution accelerator still reveals great superiority in endurance (up to 10^{12}) and feasibility in deep learning.[33]

To boost throughput, the implementation of RRAM onto SoC emphasizes localization and parallelism. In a H-tree format,[100,107] the leaves of H-tree are instructed by logic/data pairs to enable parallel processing of local data. The control bus determines in-memory computing procedures in RRAM crossbar array, which could be divided into four steps. The first logic configuration step would preprogram the required logic functionality into the RRAM crossbar with a specified pattern. The second load operand step matches up the data (weights) address with the corresponding address of logic input. The third execution step is to execute operation recurrently for a few loops. The last write-back procedure is to send the computation result back to the data array (instead of out-side CPU).

The RRAM crossbar arrays are separated into data storage units and logic operating units based on the discrepancy of functionality. The layout setting of each crossbar array varies as functionality varies. For instance, in the RRAM crossbar, the reading/writing procedures are operated row by row, whereas in the logic data RRAM crossbar, all rows are activated spontaneously to take all coming inputs. To annihilate the overhead caused by analog–digital (AD/DA) conversion, we utilize three layers of RRAM crossbars to simplify the complexity of NN operation and enable digital RRAM crossbar implementation, as shown in **Figure 6**. We proposed a novel communication protocol[100] for memory and logic operation,[108,109] which contains an instruction queue, an instruction decoder, an address decoder, and an SRAM array. As for the operation frequency inequality between the RRAM crossbar and external processor, the instructions from the outside would be held in the instruction queue temporally. The stored instruction is fed into the instruction decoder on a first-come-first-serve fashion. The address decoders are assigned with row and column indices from instruction. The computation result is stored into the SRAM array for a short period of time and written back to the data array.

The RRAM crossbar reveals great compatibility with CMOS technology in 3D integration. As shown in **Figure 7**a, the proposed 3D structure of CMOS–RRAM accelerator contains two layers of the RRAM crossbar for input buffering and logic operation, one layer of CMOS for read out, and other functionalities in machine learning, e.g., pipeline divider, look-up table (LUT) for division operation, and activation function. Figure 7b shows the design flow of the incremental machine learning procedure based on the CMOS–RRAM accelerator. To map the desired machine learning architecture onto the proposed CMOS–RRAM accelerator, the machine learning algorithm should be analyzed and modified in a trade-off between accuracy and power efficiency. The 3D architecture of CMOS–RRAM crossbar has three major merits. First, utilizing RRAM crossbar as input buffer can alleviate the leakage problem. The stacking structure of RRAM crossbar enables parallel computing from layer to layer in high throughput. Second, the RRAM crossbar array is inherently capable of performing dot product with high parallelism and power efficiency. Last but not least, the great compatibility between RRAM and CMOS enables the fast deployment of the nonlinear function and division in the machine learning algorithm.

The full-digitalized RRAM crossbar exhibits great robustness when encountering computing variance caused by the stochastic switching property.[87] However, the intermediate states are

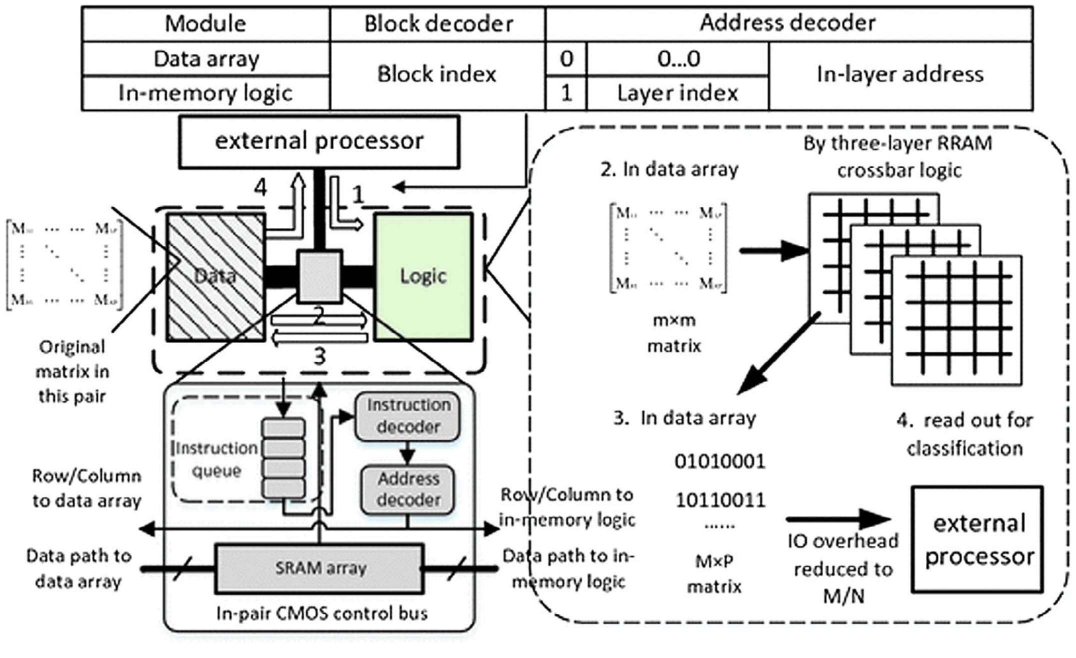


Figure 6. Detailed structure of control bus and communication protocol. Reproduced with permission.[100] Copyright 2017, Springer International Publishing AG.

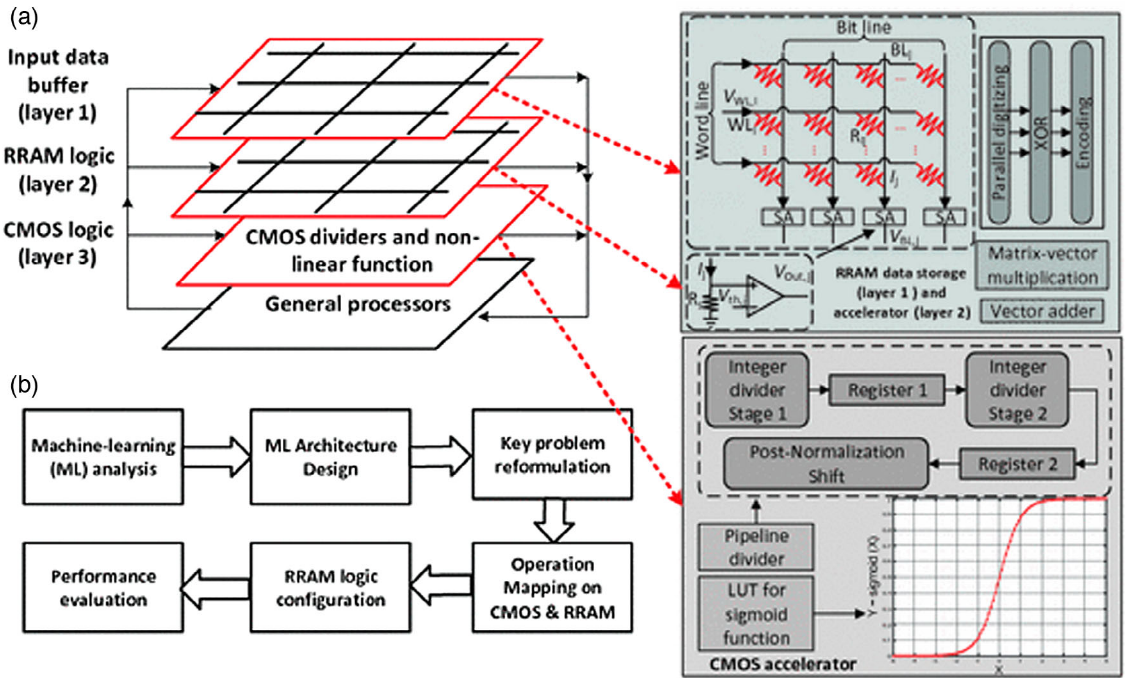


Figure 7. a) 3D stacking RRAM–CMOS accelerator architecture. b) Incremental machine learning algorithm design and mapping flow onto the proposed accelerator. Reproduced with permission.[100] Copyright 2017, Springer International Publishing AG.

much more likely to encounter resistance variance caused by the inadequate conductive filament forming, thus suffering from precision loss in computation. With a fully digitalized computing interface, the requirement for AD/DA conversion in the traditional analogy RRAM crossbar is waived.

The vector matrix multiplication contains three procedures, the parallel digitizing, the XOR operation, and the encoding. The parallel digitizing divide splits the vector matrix multiplication to multiple inner-product operations of two vectors. Every inner product is generated in the corresponding crossbar and captured by ladder-type sensing threshold voltages for each column. Based on the results of previous parallel digitizing, XOR operation is conducted for each of the two adjacent bits. The third step takes the output of the XOR step and produces inner-product results in the binary format as an encoder. The encoding layer functions as a LUT for sigmoid activation.

To achieve the neuromorphic computing model based on state-of-the-art RRAM with high parallelism and high energy efficiency, we present a binary convolution NN on the RRAM crossbar array[94] based on binarized CNN (BCNN). The CNN based on the digital RRAM crossbar has a peak throughput of 792 giga operations per second (GOPS) with a power consumption of 4.5 mW, which exhibits superiority in computing velocity and power efficiency over traditional CPU-based accelerators. BCNN is fully implemented in the RRAM crossbar, which includes bit-wise convolution, batch normalization, max pooling, and activation. The top-down overview of RRAM in-memory computing architecture is shown in **Figure 8**. In such architecture, data-logic pairs communicate through in-pair control bus, and instructions will be initially decoded/encoded in a first-come-first-serve fashion.[100] The proposed architecture enables data localization and the majority of data transmission occur within data-logic pairs; thus, I/O communication load from memory to general process can be reduced by a great portion. The convolution procedure can be divided into two steps, XNOR operation and bit-count results of two vectors, similar to the previously discussed digitized convolution protocol.[100]

The bit-wise batch normalization can be achieved using two digital RRAM crossbar arrays in implementation, as shown in **Figure 9**. The first crossbar array functions as XOR operation on adjacent bits of the output. The second RRAM works as a LUT to enable straight readout of the batch normalization result.[94] For the following procedure, we perform binarization for the batch-normalized results and then execute max pooling. The RRAM crossbar outputs 0 when all numbers are negative. If there is at least one positive number in the pooling region, the RRAM crossbar outputs a nonzero result. These bit-wise batch normalizations, max pooling, and binarization are shown in Figure 9.

The binary CNN cannot satisfy all application scenarios, as it inherently suffers from accuracy loss. When the requirement for high accuracy overwhelmed the requirement for high power efficiency, high-precision computation topology is needed. To address such problems, we present the four-bit RRAM crossbar-based-quantized deep NN (ResNet-50) protocol.[95] Our protocol cannot only guarantee a training accuracy of 88.1% (only 2.5% less than full-precision computing), but also reveal great advantages in power consumption and latency over traditional CPUs/GPUs and CMOS-application specific integrated circuit (ASIC)-based implemented accelerators.[95] To realize our multiple-bit ResNet-50 model, we exploit a four-bit RRAM model.[110] Because we are using analogue RRAM for computation, the three

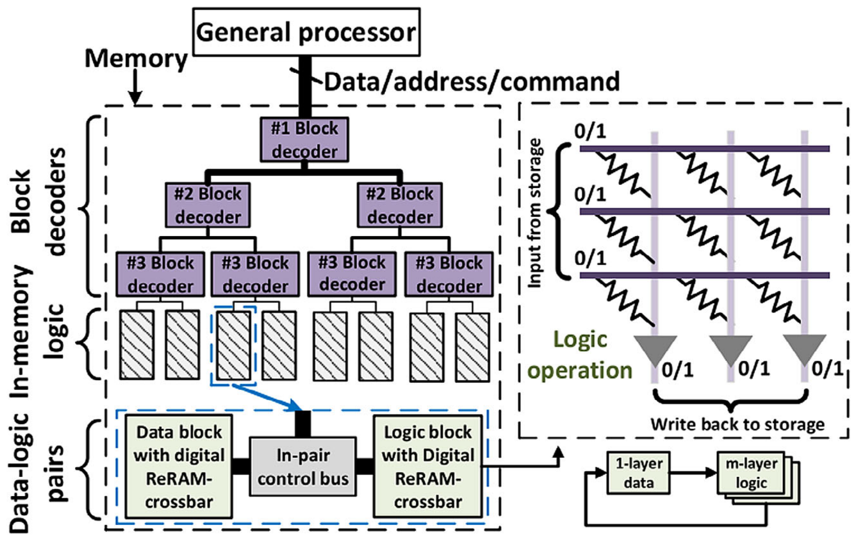


Figure 8. Overview of the digital RRAM crossbar-based in-memory computing architecture. Reproduced with permission.[94] Copyright 2017, IEEE.

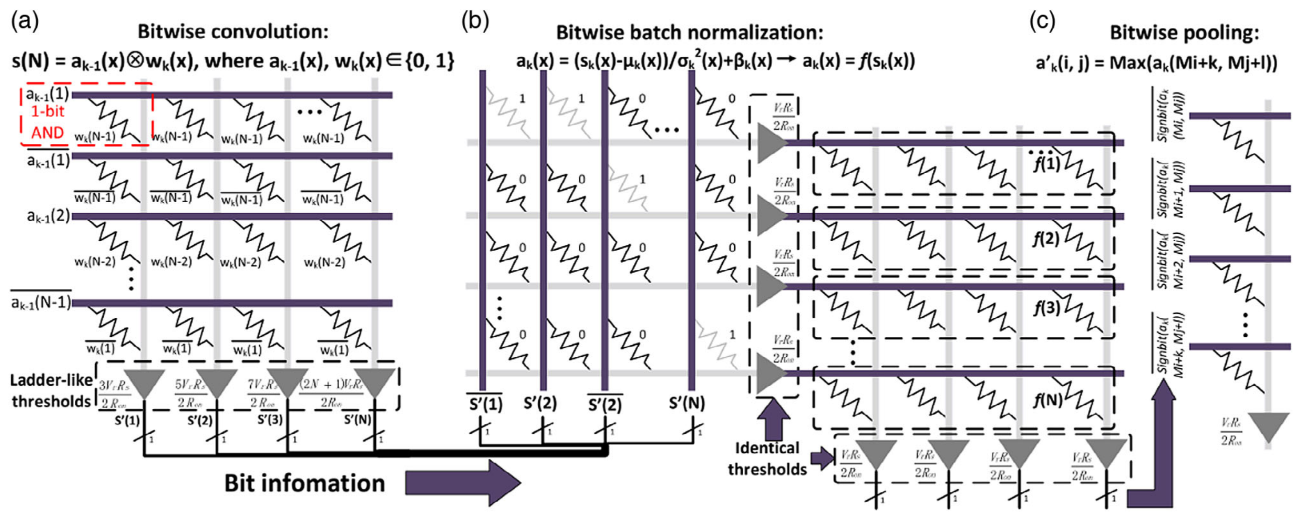

(a)
Bitwise convolution:
s(N) = a_{k-1}(x)⊗w_k(x), where a_{k-1}(x), w_k(x)∈{0, 1}
1-bit AND
Ladder-like thresholds
Bit infomation
(b)
Bitwise batch normalization:
a_k(x) = (s_k(x)-μ_k(x))/σ_k^2(x)+β_k(x) → a_k(x) = f(s_k(x))
Identical thresholds
(c)
Bitwise pooling:
a'_k(i, j) = Max(a_k(Mi+k, Mj+l))

Figure 9. Digital RRAM crossbar-based BCNN. a) Bit-wise convolution, b) bit-wise batch normalization, and c) bit-wise max pooling. Reproduced with permission.[95] Copyright 2019, Elsevier B. V.

steps for the digital RRAM convolution procedure is no longer required. Therefore, the applied input voltages versus word line will perform vector matrix multiplication automatically. The analog RRAM crossbar has a very large on–off resistance ratio when compared with the digital RRAM crossbar, which renders the analog RRAM crossbar with less sensitivity to the IR drop on a transmission wire. We exploit similar data-logic pair control strategies for our four-bit RRAM crossbar-based quantized ResNet-50, to enable local processing of data and alleviate the communication load problem between memory and general processor.[94,95,100]

Generally, the RRAM crossbar has positive values naturally. To represent the negative value in activation and convolution, we utilize adjacent WLs in the RRAM crossbar for both positive and negative computing. As shown in Figure 8, we use a $2N\times N$ RRAM crossbar for the convolution procedure. The four-bit AD/DA units convert the input/output of the RRAM crossbar into the desired feature map. The quantized weights are configured and mapped into the conductance of the RRAM crossbar on corresponding columns. In ResNet-50, we have to add the convolution output back into the input feature map. Thus, in our proposed protocol, we have shown the three-layer RRAM crossbar architecture for ResNet-50. As a result, all four-bit quantization convolution, batch normalization, activation (ReLU), and add-residue procedure can be mapped into the RRAM–CMOS format.[95]

A quantized four-bit RRAM-based ResNet-50 use RRAM[96] with 16 resistance states to enable four-bit computation, localize memory, and boost parallelism.[95] ResNet based on RRAM exhibits competitive performance in area occupation of $0.83\,\text{mm}^2$, power consumption, and efficiency (8.5 mW and 75.2 throughput [TOPs]/J).

5. System-Level Demonstration of Low-Power Neuromorphic Chips

Neuromorphic chips include coprocessors for deep learning acceleration and NN models for a computing system. The specially designed coprocessors are deployed in NNs to accelerate or optimize the deep learning procedure, e.g., general matrix–matrix multiplication (GEMM) in GPU for expediting image recognition/classification.[111] Mixed precision multiply accumulator (MAC) in GPUs or ASIC exploits computing efficiency using different message formats in different computing procedures[112,113] and calibrated PE design to localize memory and boost communication bandwidth,[114,115] e.g., lower-bit floating point 16 for multiplication and higher-bit floating point 32 or 64 for summation. The [93] field-programmable gate array (FPGA) coprocessor is largely used in deep learning applications for neuromorphic computing core modification and throughput optimization. A lot of designs have been successfully demonstrated on FPGA with outstanding improvements than concurrent neuromorphic design, e.g., systolic array for filtering convolution[116] on robotic autopilot, reconfigurable architecture for performance enhancement and bandwidth extension,[117] and downscaling of bandwidth requirement to release the constraint on data transmission.[118]

The system-level neuromorphic computing chips are now available. The SNN algorithms are largely implemented in hardware, e.g., Loihi of Intel, TrueNorth of IBM, and Neurogrid of Stanford University. All of these system-level neuromorphic chips use SRAM/DRAM to store synaptic weights and construct LUT for neuromorphic implementation. For instance, TrueNorth[6] utilizes capacitors, 12-transistor (12 T) SRAM, 6-transistor (6 T) SRAM for spike trains scheduling and synaptic event data memory, respectively. The size of SRAM shrinks along with the CMOS scaling node technologies, and a standard 6 T SRAM occupies 0.152 μm^2 of on-chip area.[119,120] As shown in **Table 1**, the SRAM implementation strategies of TrueNorth and NeuroGrid have the advantages of low power consumption, low latency, and high area efficiency. However, as SRAM is constructed by several transistors (bistable triggering circuit) for data storage, the bitcell density is inevitably large ($\approx 150\ F^2$) and further constrain the expansion of on-chip memory capability (several to tens of MB), which renders SRAM incompetent for deep learning, especially for dealing with an enormous amount of parameters. Another obvious drawback of SRAM is the leakage problem induced by its multiple-transistor structures. The traditional SRAM still suffers from low parallelism for its row-by-row operation.[121] A novel layout of SRAMs is proposed to boost parallel processing, which is similar to the layout of aforementioned NVM in-memory computing array that uses analog signal to conduct MAC operation.[122–125] To achieve such high parallelism, additional transistors per bitcell are required, which further exacerbate the aforementioned density and leakage dilemmas. Generally, the NVM architecture[126] (1T1R, 2T2R, 2M) has great advantages in density, especially for 3D-stacked configuration. Its less-transistor design makes the NVM array much more robust to IR drop and leakage issues.

The TrueNorth chip was designed by IBM for developing a non-Von Neumann neuromorphic chip with the characteristics of low power, real-time operation, and scalable and default tolerance.[9] The TrueNorth chip contains a mixed asynchronous–synchronous design approach, in which a synchronous circuit is used for uniform clock distribution, and an asynchronous circuit determines the ON/OFF states of the synchronous circuit. As a fully digitalized SoC, TrueNorth contains 4096 neurosynaptic cores, which are arranged in a 2D array with an event-driven structure and routed in mesh distribution. The chip carries 1 million neurons and 256 million synapses with a peak computational performance of 58 GOPS and computational energy efficiency of 400 GSOPS per watt (GSOPS/W).[9,24] The TrueNorth chip was simulated to have 65 mV per core, when operating at 0.75 V with a firing rate of 20 Hz.

The Neurogrid chip, a mixed analog–digital neuromorphic chip, aims to detect real-time communication in biological neurons by capturing the electrical dynamics among each bioinspired electronic unit.[16] The system contains 16 chips wired by a tree router[127] on the daughterboard's FPGA,[16] which uses multicast broadcasting and a local analog diffuser to transmit messages asynchronously among each neuron.

Table 1. Implementation and optimization strategies of TrueNorth and NeuroGird as well as their performance comparison.

Strategy	TrueNorth	NeuroGrid	Performance
Clock distribution	An asynchronous circuit design to minimize idle power consumption and optimize area efficiency. A global system synchronization to enable real-time operation.	Asynchronous design.	Asynchronous circuit design has advantages in power efficiency, throughput, and area efficiency, but lack of time efficiency in real-time tasks, when compared with synchronous design. A well-balanced mixed synchronous–asynchronous design could mitigate the drawbacks of both designs, while boost the good.
Memory localization and routing	2D mesh communication network enables each router with six independent processing procedures. The package delivered to each router was transmitted via one of its four nearest neighbors or its localized memory.	H-tree router with four merges and three splits datapath, which is divided into six bit pairs to boost power efficiency. For lack of embedded memory, the delivery of package is done without any lookups.	The hierarchical design of 2D mesh network routing enables fan-out of single package toward multiple recipients' local memory, which could reduce the frequency of off-chip data fetching and is beneficial in real-time task.
Neuromorphic core	SNN model in CMOS–ASIC (SRAM) implementation.	SNN model in FPGA (SRAM) implementation.	Locally executed neuron core in TrueNorth enables better scalability, for the clock skew problem in the long range implementation.

The Neurogrid possesses over 65 000 neurons and 500 million synapses. Every chip is fabricated in 180 nm CMOS node technology, and the neurons on the same cortex layer (chip) show a characteristic of congruity. The Neurogrid chip was emulated to have 3.1 watt of power consumption on daughterboard's FPGA or 941 pJ per synaptic event with over 7980 synapses connected to each neuron.

All modern processors are based upon finite-state automatons, which transit among a finite number of predefined states. These states are defined as digital logic in silicon and impossible to be modified once generated. A large variety of neuron activities, such as learning, functioning, or developing, require a comprehensive understanding of neuron methodology. Unlike digital signal, analog signal is pervasive in the real world, such as sound, shake, and the signal travel through the neuron system. The analog part of the neuron model has advantages in simplicity and precision in biosynaptic behavior description, whereas the digital part of the neuron model guarantees the accuracy of the system.

6. Conclusions

In the current context of artificial intelligence and big data, various application scenarios impose great challenges to the efficiency of computing. Low-power computing by neuromorphic engineering is a promising way for expediting the engineering deployment of AI into daily life, but also enhancing our understanding on brain functionality. We started with the biomimic NVM devices of STT-MRAM, PCM, and RRAM, which enable in-memory computing, fast programming, e.g., sub-ns writing speed and one-step matrix multiplication, and low operation power especially idle power. Analog data processed in these NVMs enable the designed chip with great advantages in terms of power efficiency and latency. Due to easy integration with CMOS, the digital data processed in the peripheral circuit guarantees robustness and parallelism. Thus, these NVM devices are utilized as synapse (processor) and memory in computation simultaneously and surpass traditional Von Neumann architecture. However, these NVMs still suffer from problems associated with endurance, device variation, yield, and speed, which hinder NVMs from large-scale commercialization. The overhead problems caused by AD/DC conversion in NVM arrays still dominate the area occupation and power consumption in neuromorphic chips. To achieve neuromorphic computing with desired performance, the trade-off between latency and power consumption of the peripheral circuit (AD/DC) must be carefully accessed. We also described the architecture of NN, the SNN, CNN, and the training algorithm for SNN and CNN. The event-driven SNN exploits temporal coding STDP and therefore is naturally more friendly (power consumption and reaction duration) in real-time application than the traditional rate-driven system. We then pointed out the current dilemma of SNNs, such as lack of specific training dataset for the event-driven system and proper training methodologies for multilayer training. Therefore, we presented the infused spiking CNN and gradient-based SNN. The property of the NVM array in matrix multiplication has led to a great amount of applications onto convolution NN. The future of neuromorphic computing is promising. But it still requires great endeavor to enhance the stability and consistency of NVM devices. For algorithm implementation, there is still great demand for high fault tolerance, quick convergence, and application-specialized algorithm for various types of NNs. Overall, low-power computing with neuromorphic engineering requires interdisciplinary intelligence to integrate devices, circuits, architectures, and algorithms with desired functionalities.

Acknowledgements

This work was supported by the Research Grant Council of Hong Kong (PolyU 152053/18E), the Science and Technology Innovation Committee Foundation of Shenzhen (grant nos. JCYJ20180504165652917 and JCYJ20180504170454381), and Key-Area Research and Development Program of Guangdong Province (grant no. 2019B010142001).

Conflict of Interest

The authors declare no conflict of interest.

Keywords

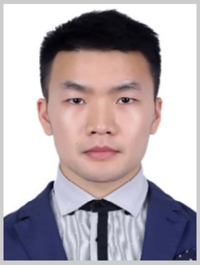
in-memory computing, low power neuromorphic computing, nonvolatile memories, synaptic devices

Received: June 30, 2020
Revised: September 12, 2020
Published online: December 7, 2020

[1] E. Ayanoglu, IEEE ComSoc Technical Committees Newsletter, IEEE TCN November **2019**.
[2] C. Zhou, X. Wang, S. Raju, Z. Lin, D. Villaroman, B. Huang, H. L.-W. Chan, M. Chan, Y. Chai, *Nanoscale* **2015**, *7*, 8695.
[3] S. Kumar, J. P. Strachan, R. S. Williams, *Nature* **2017**, *548*, 318.
[4] S. P. Lau, L.-J. Li, Y. Chai, *Adv. Funct. Mater.* **2017**, *27*, 1701403.
[5] E. C. Ahn, H. S. P. Wong, E. Pop, *Nat. Rev. Mater.* **2018**, *3*, 18009.
[6] H. S. P. Wong, S. Salahuddin, *Nat. Nanotechnol.* **2015**, *10*, 191.
[7] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, A. Leblanc, *IEEE J> Solid-State Circuits* **1974**, *9*, 256.
[8] W. A. Wulf, S. A. Mckee, *ACM Sigarch Computer Arch. News* **1995**, *23*, 20.
[9] F. Akopyan, J. Sawada, A. S. Cassidy, R. Alvarezicaza, J. V. Arthur, P. A. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1537.
[10] J. Hennessy, D. Patterson, K. Asanovic, Computer architecture: a quantitative approach, Sixth edition, Morgan Kaufmann Publishers Cambridge, MA **2019**.
[11] M. J. Flynn, *Proc. IEEE* **1966**, *54*, 1901.
[12] J. Frascaroli, S. Brivio, E. Covi, S. Spiga, *Sci. Rep.* **2018**, *8*, 7178.
[13] C. Amza, A. L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, W. Yu, W. Zwaenepoel, *IEEE Computer* **1996**, *29*, 18.
[14] D. Ielmini, H. S. P. Wong, *Nat. Electr.* **2018**, *1*, 333.
[15] M. E. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. D. Dimou, P. Joshi, N. Imam, S. Jain, *IEEE Micro* **2018**, *38*, 82.
[16] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, K. Boahen, *Proc. IEEE* **2014**, *102*, 699.
[17] Y. Chai, *Nature* **2020**, *579*, 32.

[18] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, S. Millner. in *Int. Symp. on Circuits and Systems* **2010**.
[19] G. H. Loh, in *Int. Symp. on Computer Architecture* **2008**.
[20] R. Strenz, in *Int. Electron Devices Meeting* **2011**.
[21] R. C. Merkle, Foresight Institute **1989**, Foresight Update 6.
[22] S. B. Laughlin, R. R. D. R. Van Steveninck, J. Anderson, *Nat. Neurosci.* **1998**, *1*, 36.
[23] Z. Jiang, Y. Wu, S. Yu, L. Yang, K. Song, Z. Karim, H. S. P. Wong, *IEEE Trans. Electron Devices* **2016**, *63*, 1884.
[24] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, D. S. Modha, *Science* **2014**, *345*, 668.
[25] 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, J. P. Strachan, *Adv. Mater.* **2018**, *30*, 1705914.
[26] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, R. S. Williams, in *Design Automation Conf.* **2016**.
[27] S. Yu, *Proc. IEEE* **2018**, *106*, 260.
[28] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, H. Li, H. Wu, R. Zhao, Y. Xie, L. Shi, *Nature* **2019**, *572*, 106.
[29] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, H. Qian, *Nature* **2020**, *577*, 641.
[30] K. Roy, A. Jaiswal, P. Panda, *Nature* **2019**, *575*, 607.
[31] A. Sherstinsky, *Phys. D: Nonlinear Phenomena* **2020**, *404*, 132306.
[32] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, *Nature* **2016**, *529*, 484.
[33] S. Yu, Resistive Random Access Memory (RRAM): From Devices to Array Architectures (Ed.: K. Iniewski) Morgan & Claypool **2016**.
[34] F. Zhou, Z. Zhou, J. Chen, T. H. Choy, J. Wang, N. Zhang, Z. Lin, S. Yu, J. Kang, H. S. P. Wong, Y. Chai, *Nat. Nanotechnol.* **2019**, *14*, 776.
[35] F. Zhou, J. Chen, X. Tao, X. Wang, Y. Chai, *Research* **2019**, *2019*, 9490413.
[36] F. Zhou, Y. Liu, X. Shen, M. Wang, F. Yuan, Y. Chai, *Adv. Funct. Mater.* **2018**, *28*, 1800080.
[37] L. Chua, *IEEE Trans. Circuit Theory* **1971**, *18*, 507.
[38] H. Yu, Y. Wang, *Design Exploration of Emerging Nano-scale Non-volatile Memory*, Springer, New York, NY **2014**, pp. 45–83.
[39] D. Kuzum, S. Yu, H. P. Wong, *Nanotechnology* **2013**, *24*, 382001.
[40] D. S. Jeong, I. Kim, M. Ziegler, H. Kohlstedt, *RSC Adv.* **2013**, *3*, 3169.
[41] D. S. Jeong, K. Kim, S. Kim, B. J. Choi, C. S. Hwang, *Adv. Electron. Mater.* **2016**, *2*, 1600090.
[42] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, H. P. Wong. in *Int. Electron Devices Meeting* **2012**.
[43] K. Kim, S. Gaba, D. C. Wheeler, J. Cruzalbrecht, T. Hussain, N. Srinivasa, W. Lu, *Nano Lett.* **2012**, *12*, 389.
[44] J. Joshua Yang, M. X. Zhang, M. D. Pickett, F. Miao, J. Paul Strachan, W.-D. Li, W. Yi, D. A. A. Ohlberg, B. Joon Choi, W. Wu, J. H. Nickel, G. Medeiros-Ribeiro, R. Stanley Williams, *Appl. Phys. Lett.* **2012**, *100*, 113501.
[45] 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, in *Int. Electron Devices Meeting* **2012**.
[46] S. Ali, J. Bae, C. H. Lee, S. Shin, N. P. Kobayashi, *Org. Electron.* **2017**, *41*, 73.
[47] C. Wu, T. W. Kim, T. Guo, F. Li, D. U. Lee, J. J. Yang, *Adv. Mater.* **2017**, *29*, 1602890.
[48] H. Huang, L. Ni, K. Wang, Y. Wang, H. Yu, *IEEE Trans. Nanotechnol.* **2018**, *17*, 645.
[49] A. V. Khvalkovskiy, D. Apalkov, S. Watts, R. Chepulskii, R. S. Beach, A. Ong, X. Tang, A. Driskill-Smith, W. H. Butler, P. B. Visscher, D. Lottis, E. Chen, V. Nikitin, M. Krounbi, *J. Phys. D: Appl. Phys.* **2013**, *46*, 074001.
[50] D. Apalkov, A. V. Khvalkovskiy, S. Watts, V. Nikitin, X. Tang, D. K. Lottis, K. Moon, X. Luo, E. Chen, A. E. Ong, *ACM J. Emerging Technol. Comput. Syst.* **2013**, *9*, 13.
[51] Y. Wang, Y. Shang, H. Yu, in *Conf.: IEEE Non-Volatile Memory Technology Symp.* **2012**.
[52] S. Yang, F. Wei, Y. Hao, in *17th Asia and South Pacific Design Automation Conf.* **2012**.
[53] F. Yuan, Z. Zhang, C. Liu, F. Zhou, H. M. Yau, W. Lu, X. Qiu, H. S. P. Wong, J. Dai, Y. Chai, *ACS Nano* **2017**, *11*, 4097.
[54] Z. Diao, D. Apalkov, M. Pakala, Y. Ding, A. Panchula, Y. Huai, *Appl. Phys. Lett.* **2005**, *87*, 232502.
[55] D. Kuzum, R. Jeyasingh, B. Lee, H. S. P. Wong, *Nano Lett.* **2012**, *12*, 2179.
[56] G. Indiveri, E. Chicca, R. J. Douglas, *IEEE Trans. Neural Networks* **2006**, *17*, 211.
[57] R. Waser, R. Dittmann, G. Staikov, K. Szot, *Adv. Mater.* **2009**, *21*, 2632.
[58] Y. Chai, Y. Wu, K. Takei, H. Chen, S. Yu, P. C. H. Chan, A. Javey, H. P. Wong, *IEEE Trans. Electron Devices* **2011**, *58*, 3933.
[59] S. Fujii, J. A. C. Incorvia, F. Yuan, S. Qin, F. Hui, Y. Shi, Y. Chai, M. Lanza, H. P. Wong, *IEEE Electron Device Lett.* **2018**, *39*, 23.
[60] P. Chen, B. Lin, I. Wang, T. Hou, J. Ye, S. Vrudhula, J. Seo, Y. Cao, S. Yu, in *IEEE/ACM Int. Conf. on Computer-Aided Design* **2015**.
[61] J. Liu, B. Jaiyen, R. Veras, O. Mutlu, *39th Annual Int. Symp. on Computer Architecture*, Association for Computing Machinery, New York, NY **2012**, Vol. *40*, pp. 1–12.
[62] I. T. Wang, C.-C. Chang, L.-W. Chiu, T. Chou, T.-H. Hou, *Nanotechnology* **2016**, *27*, 365204.
[63] K. He, X. Zhang, S. Ren, J. Sun, in *Computer Vision and Pattern Recognition* **2016**.
[64] Y. Wang, X. Li, K. Xu, F. Ren, H. Yu, *IEEE Trans. Biomed. Circuits Syst.* **2017**, *11*, 255.
[65] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, New York, NY **1995**.
[66] A. L. Hodgkin, A. F. Huxley, *J. Physiol.* **1952**, *117*, 500.
[67] E. M. Izhikevich, *IEEE Trans. Neural Networks* **2003**, *14*, 1569.
[68] Y. Liu, X. Wang, *J. Comput. Neurosci.* **2001**, *10*, 25.
[69] A. S. Cassidy, P. A. Merolla, J. V. Arthur, S. K. Esser, B. L. Jackson, R. Alvarezicaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, in *Int. Joint Conf. on Neural Network* **2013**.
[70] G. Bi, M. Poo, *J. Neurosci.* **1998**, *18*, 10464.
[71] M. Hu, Y. Chen, J. J. Yang, Y. Wang, H. H. Li, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2017**, *36*, 1353.
[72] S. Yin, X. Sun, S. Yu, J. Seo, arXiv: Emerging Technologies **2019**.
[73] J. Deng, W. Dong, R. Socher, L. Li, K. Li, L. Feifei, in *Computer Vision and Pattern Recognition* **2009**.
[74] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, arXiv: Neural Evolutionary Computing **2012**.
[75] R. Van Rullen, S. J. Thorpe, *Neural Comput.* **2001**, *13*, 1255.
[76] F. Ponulak, A. Kasinski, *Neural Comput.* **2010**, *22*, 467.
[77] S. M. Bohte, J. N. Kok, J. A. H. La Poutre, *Neurocomputing* **2002**, *48*, 17.
[78] A. Sengupta, Y. Ye, R. Wang, C. Liu, K. Roy, *Front. Neurosci.* **2019**, *13*, 95.
[79] Y. Cao, Y. Chen, D. Khosla, *Int. J. Comput. Vision* **2015**, *113*, 54.

[80] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. Liu, M. Pfeiffer, in *Int. Joint Conf. on Neural Network* **2015**.
[81] E. Hunsberger, C. Eliasmith, cs.LG arXiv, 1510.08829 **2015**.
[82] J. H. Lee, T. Delbruck, M. Pfeiffer, *Front. Neurosci.* **2016**, *10*, 508.
[83] F. Zenke, S. Ganguli, *Neural Comput.* **2018**, *30*, 1514.
[84] P. U. Diehl, M. Cook, *Front. Comput. Neurosci.* **2015**, *9*, 99.
[85] T. Masquelier, R. Guyonneau, S. J. Thorpe, *Neural Comput.* **2009**, *21*, 1259.
[86] T. Masquelier, R. Guyonneau, S. J. Thorpe, *Plos One* **2008**, *3*, e1377.
[87] T. Masquelier, S. R. Kheradpisheh, *Front. Comput. Neurosci.* **2018**, *12*, 1662.
[88] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, *Proc. IEEE* **1998**, *86*, 2278.
[89] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. J. Mollura, R. M. Summers, *IEEE Trans. Med. Imag.* **2016**, *35*, 1285.
[90] X. Zeng, S. Wen, Z. Zeng, T. Huang, *Neural Comput. Appl.* **2018**, *30*, 503.
[91] C. Yakopcic, M. Z. Alom, T. M. Taha, *Int. Joint Conf. on Neural Networks* **2016**, pp. 963–970.
[92] Y. Wang, L. Xia, T. Tang, B. Li, S. Yao, M. Cheng, H. Yang, in *IEEE Int. Symp. on Circuits and Systems* **2016**.
[93] Y. Li, Z. Liu, K. Xu, H. Yu, F. Ren, *ACM J. Emerg. Technol. Comput. Syst.* **2018**, *14*, 18.
[94] L. Ni, Z. Liu, H. Yu, R. V. Joshi, *IEEE J. Exploratory Solid-State Computat. Devices Circuits* **2017**, *3*, 37.
[95] Y. Cheng, C. Wang, H. Chen, H. Yu, *Integration* **2019**, *69*, 345.
[96] M. Hu, H. Li, Q. Wu, G. S. Rose, in: *Design Automation Conf.* **2012**.
[97] A. E. Bryson, W. F. Denham, *J. Appl. Mech.* **1962**, *29*, 247.
[98] A. E. Bryson, Y. C. Ho, Applied Optimal Control: Optimization, Estimation and Control Blaisdell Pub Waltham, MA **1969**.
[99] P. J. Werbos, *System Modeling and Optimization. Lecture Notes in Control and Information Sciences*, (Eds: R. F. Drenick, F. Kozin), Vol. *38*, Springer, Heidelberg **1982**.
[100] H. Yu, L. Ni, H. Huang, *Computational Intelligence* (Eds: S. Vaidyanathan, C. Volos) **2017**, p. 701.
[101] J. F. Kang, P. Huang, R. Han, Y. Xiang, X. Cui, X. Y. Liu, in *Int. Conf. on ASIC* **2019**.
[102] R. Han, P. Huang, Y. Xiang, C. Liu, Z. Dong, Z. Q. Su, Y. B. Liu, L. Liu, X. Liu, J. Kang, in *Int. Symp. on Circuits and Systems* **2018**.
[103] X. Guo, F. M. Bayat, M. Prezioso, Y. Chen, B. Nguyen, N. Do, D. B. Strukov. in *Custom Integrated Circuits Conf.* **2017**.
[104] Y. Xiang, J. F. Kang, P. Huang, H. Z. Yang, K. L. Wang, R. Han, W. Shen, Y. Feng, C. Liu, X. Y. Liu, in *Int. Electron Devices Meeting* **2019**.
[105] R. Bez, E. Camerlenghi, A. Modelli, A. Visconti, *Proc. IEEE* **2003**, *91*. 489.
[106] F. M. Bayat, X. Guo, M. Klachko, N. Do, K. K. Likharev, D. B. Strukov, in *Device Research Conf.* **2016**.
[107] Y. Wang, H. Yu, L. Ni, G. Huang, M. Yan, C. Weng, W. Yang, J. Zhao, *IEEE Trans. Nanotechnol.* **2015**, *14*, 998.
[108] D. Xu, N. Yu, H. Huang, P. D. S. Manoj, H. Yu, *IEEE Design Test* **2018**, *35*, 91.
[109] P. D. S. Manoj, J. Lin, S. Zhu, Y. Yin, X. Liu, X. Huang, C. Song, W. Zhang, M. Yan, Z. Yu, *IEEE Trans. Circuits Syst. I-Regular Pap.* **2017**, *64*, 1432.
[110] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, Q. Xia, *Nat. Electron.* **2018**, *1*, 52.
[111] Y. Chen, H. Li, C. Wu, C. Song, S. Li, C. Min, H.-P. Cheng, W. Wen, X. Liu, *Integration* **2018**, *61*, 49.
[112] E. Lindholm, J. R. Nickolls, S. Oberman, J. Montrym, *IEEE Micro* **2008**, *28*, 39.
[113] D. I. Lyakh, *Comput. Phys. Commun.* **2015**, *189*, 84.
[114] Y. Chen, J. Emer, V. Sze, in *Int. Symp. on Computer Architecture* **2016**.
[115] Y. Chen, T. Krishna, J. Emer, V. Sze, in *Int. Solid-State Circuits Conf.* **2016**.
[116] S. Sirowy, A. Forin, Where's the beef? Why FPGAs are so fast, Microsoft Research, Microsoft Corp **2008**.
[117] S. Chakradhar, M. Sankaradas, V. Jakkula, S. Cadambi. in *Int. Symp. on Computer Architecture* **2010**.
[118] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, J. Cong, in *Field Programmable Gate Arrays* **2015**.
[119] Y. Shin, K. Shin, P. Kenkare, R. Kashyap, H. Lee, D. Seo, B. Millar, Y. Kwon, R. Iyengar, M. Kim, in *Int. Solid-State Circuits Conf.* **2013**.
[120] M. Bohr, I. A. Young, *IEEE Micro* **2017**, *37*, 20.
[121] S.-Y. Wu, C. Lin, M. Chiang, J. Liaw, J. Cheng, S. Yang, C. Tsai, P. Chen, T. Miyashita, C. Chang, V. S. Chang, K. Pan, J. Chen, Y. Mor, K. Lai, C. Liang, H. F. Chen, S. Chang, C. Lin, S. Jang, *IEEE Int. Electron Devices Meeting* **2016**, pp. 2.6.1–2.6.4.
[122] A. Biswas, A. P. Chandrakasan, in *Int. Solid-State Circuits Conf.* **2018**.
[123] W. Khwa, J. Chen, J. Li, X. Si, E. Yang, X. Sun, R. Liu, P. Chen, Q. Li, S. Yu, in *Int. Solid-State Circuits Conf.* **2018**.
[124] Z. Jiang, S. Yin, M. Seok, J. Seo, in *Symp. on VLSI Technology* **2018**.
[125] H. Valavi, P. J. Ramadge, E. Nestler, N. Verma, in *Symp. on VLSI Circuits* **2018**.
[126] O. Krestinskaya, A. P. James, L. O. Chua, *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *31*, 4.
[127] P. A. Merolla, J. V. Arthur, R. Alvarez, J. Bussat, K. Boahen, *IEEE Trans. Circuits Syst.* **2014**, *61*, 820.



Dingbang Liu received his bachelor's and master's in electrical engineering and electrical and computer engineering from the State University of New York at Binghamton. Now, he is pursuing his Ph.D. in Hong Kong Polytechnic University. His current research interests focus on neuromorphic chips and RISC-V SoC.

Hao Yu is a professor at the Southern University of Science and Technology. He received his B.S. from Fudan University and Ph.D. from the Department of Electrical Engineering, University of California at Los Angeles. His current research interests include CMOS emerging technology for data sensors, links, and accelerators.

Yang Chai is an associate professor at the Hong Kong Polytechnic University. He is a member of Hong Kong Young Academy of Sciences and the vice president of Physical Society of Hong Kong. His current research interest includes low-dimensional materials for electron device applications.