Memristive devices for brain-inspired computing

Editors: Sabina Spiga, Abu Sebastian, Damien Querlioz, Bipin Rajendran

Chapter 18

System-level integration in neuromorphic co-processors

Giacomo Indiveri, Bernabe Linares Barranco, and Melika Payvand

Abstract

In this chapter we present results on system-level integration of memristive devices with neuromorphic circuits and systems. Specifically, we present an overview of the current state-of-the-art hybrid memristive-CMOS mixed-signal neuromorphic circuits for learning and plasticity and present perspectives towards integration of memristive devices in neuromorphic spiking neural network architectures. We focus on neuromorphic circuits and architectures that allow for a relatively natural integration of memristive devices, irrespective of the specific characteristics of the specific memristive device technology adopted. We address the co-integration of memristive devices with on-chip learning mechanisms, using both analog and digital CMOS circuits, to build a solid background of the functionality of neuromorphic circuits explaining how memristive devices can be implemented on them. Furthermore, we also address the system-level integration of such architectures in multi-core and multi-chip systems, for connecting them to input and output devices, such as sensors, actuators, and conventional CMOS processing devices.

18.1 Neuromorphic computing

Neuromorphic computing systems typically comprise neuron and synapse circuits arranged in a massively parallel manner to support the emulation of large-scale spiking neural networks. Different approaches have been proposed for implementing hardware implementations of neuromorphic computing systems, ranging from digital CMOS ones based on synchronous [1] and asynchronous logic [2, 3], to analog and mixed-signal ones based on standard strong inversion circuits [4] and weak-inversion circuits [5–7]. Although implemented in pure conventional CMOS technology, most of these neuromorphic architectures are optimally suited for co-integration with memristive devices, which can be used to both emulate synaptic function and to support non-volatile local storage of network parameters [8, 9]. In such architectures memory elements (e.g., that store the synaptic state) are used also as computing elements (i.e., that convert input pulses into weighted synaptic currents) and are placed just next to the main processing units (i.e., the neurons that integrate all synaptic inputs and produce output spikes). These architectures are radically different from the ones based on the classical von Neumann computer one, in which memory and compute elements are implemented in separate and distinct blocks that exchange data across a common shared bus, as quickly as possible. The spiking neural networks implemented by the neuromorphic architectures can be configured to carry out multiple types of signal processing tasks, ranging from sensory signal processing [10] to pattern recognition [6], to finite-state-machine like computation [11]. These spiking neural networks represent new brain-inspired computing paradigms and the hardware architectures that implement them have the potential of solving the von Neumann memory bottleneck problem [12] efficiently [13]: given their co-localization of memory and computation, no fast exchange of data across different memory and compute blocks takes place. In addition, these architectures can minimize power consumption by performing data-driven computation (i.e., carrying out computations only when there is data to drive the circuits), and processing the data at a rate that matches the time constants of the input signals and the real-time requirements of the task at hand. Setting the time constants of the processing elements to match those of the signals that need to be processed can reduce the power consumption and data bandwidth requirements by orders of magnitude, compared to synchronous clock-driven conventional computer approaches. Despite the use of slow, reduced precision, and variable computational elements, these architectures can achieve fast, robust, and reliable computation by virtue of their massively parallel mode of operation. Given this design strategy, these architectures can also exhibit remarkable fault-tolerance features, by taking advantage of the inherent redundancy in the use of their components. Indeed, while the memory-related constraints of conventional computers require high-speed data transfers using reliable bit-precise devices, these brain inspired computing systems, as well as the biological nervous systems they emulate, are able to perform fast and robust computation, using memory and computing elements that are slow, in-homogeneous, stochastic and faulty [8, 14, 15].

18.2 Integrating memristive devices as synapses in neuromorphic computing architectures

Typically, in neuromorphic computing architectures, very large arrays of synaptic elements are connected to a smaller number of neuron circuits. In addition to being compatible with designs that faithfully model biological neural networks, architectures comprising neurons connected to a large number of many parallel synapses support the implementation of a wide range of spiking neural network topologies, including multi-layer or deep neural networks, and recurrent neural networks. In order to configure the desired network topology and program the connectivity among neurons, neuromorphic systems typically assign an address to each neuron and encode the spikes produced by such neurons as "address-events". In these systems information is typically encoded in the timing of the address-events. Specifically, the interval between successive address-events produced by the same neuron (i.e., the inter-spike interval) can be used to represent analog values. Neural processing of analog variables can be achieved by using the digital address-event representation (AER) [16–19] and connecting multiple neurons among each other with different types of AER connectivity schemes. Spikes produced by source neurons are transmitted to one or more destination synapse circuits that integrate them with different gain factors and convey them to the post-synaptic neuron. Unlike classical digital logic circuits, these networks are typically characterized by very large fan-in and fan-out numbers. For example, in cortical networks neurons project on average to about 10'000 destinations. The type of processing and functionality of these spiking neural networks is determined by the their specific structure and parameters, such as the properties of the neurons or the weights of the synapses [20]. It is therefore important to design neuromorphic computing platforms that can be configured to support the construction of different network topologies, with different neuron and synapse properties. This requires the development of both configurable neuron/synapse circuits, and of programmable event-based routing and communication schemes. The latter elements are particularly important, because the scalability of neuromorphic systems is mainly restricted by communication requirements. Figures 18.1 and 18.2 show examples of architectures that follow two complementary approaches for achieving the implementation of large-scale neural networks. Each of the populations of neurons shown in these figures could be implemented in a single "core" and multi-population (e.g. multi-layer) networks can be implemented by designing multi-core neuromorphic processors [1–3, 7].

The high-density multi-core approach of Fig. 18.1 aims to capitalize on the nano-scale size of memristive devices and uses them as simple synaptic elements in dense cross-bar arrays [21–23]. In this approach the single synapse element is kept as simple and compact as possible using either one single passive memristive device element per synapse arranged in "1R" cross-bar arrays, or single memristive devices connected to a "select" transistor or "selector" device in "1T-1R" cross-bar arrays (see also Chapter **[XBAR CHAPTER]**). The state of a specific memristive synapse, corresponding to its synaptic weight, can be changed by appropriately setting the voltage values of the row and column lines connected to the top and bottom electrodes of the target device (e.g., with a large voltage difference ΔV), while setting the ΔV of all other

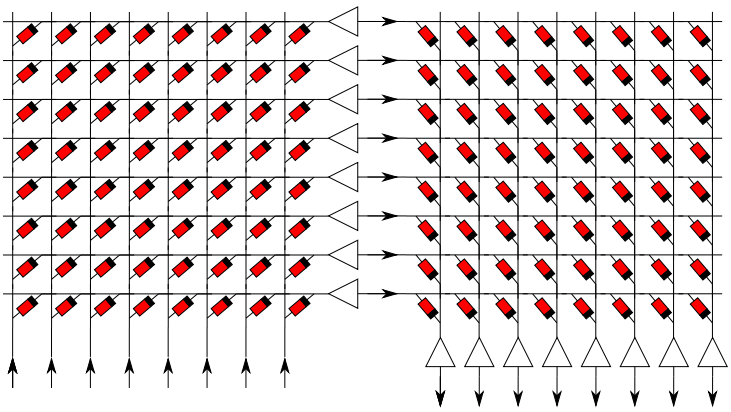

Figure 18.1: Example of cross-bar multi-neuron architecture.

devices to a low value (or ideally zero). Programming circuits that implement these operations can be designed and placed at the periphery of the cross-bar arrays. Similarly, these programming blocks can be driven by spike-based learning circuits which evaluate the state of the input and output pulses and produce an appropriate voltage profile to be applied between the top and bottom electrodes of the addressed synapse. While this approach has the advantage of being able to support very dense cross-bar array fabrications, it has the disadvantage that it requires large overhead circuitry at the periphery for the address encoders, decoders, programming logic, and voltage drivers. Depending on the technology node used, these peripheral circuits can be made quite small. However, since the technology has to support sufficiently large voltage values for forming the memristive devices and updating their state, the corresponding minimum feature size is typically not very small and the overall area used by the overhead circuitry can become prohibitive for typical neuromorphic computing designs. Another disadvantage of this approach, in case on-chip learning features are desired, is given by the fact that learning and weight updates require the cross-bar row and column select lines to be actively driven for the full duration of the memristive device set operation, which is determined by the length of the ΔV pulse used to change the memristor state. Therefore the bandwidth of the data flow in these architectures is limited by the minimum duration of the ΔV pulse. Since typical spike-based learning protocols require pre-synaptic input pulses to overlap with post-synaptic output pulses in order to produce the right ΔV pulse, these operations can keep the row and column select lines busy for very long periods, reaching even milliseconds (e.g., for learning protocols that model real synapses and/or that are used for processing sensory signals), and severely limit the overall system bandwidth, as well as power consumption.

Conversely, the multi-core approach of Fig. 18.2 forgoes the attempt to maximise density at the cross-bar level to allow the use of larger synapse blocks at the benefit of adding additional complexity within each synapse and enabling more sophisticated and massively parallel computations. Examples of more complex "compound" synapses that comprise multiple memristive devices per synapse have been shown to enable more precise modulation of

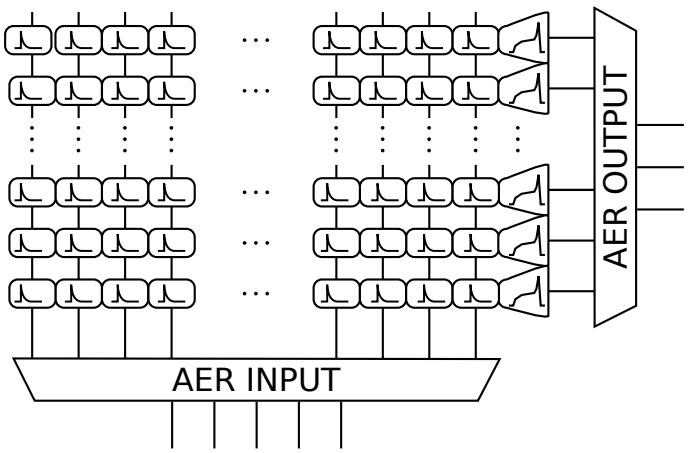

Figure 18.2: Example of AER multi-neuron architecture.

the synaptic weight over a wide dynamic range [24] and to reduce the effect of device variability [9] (e.g., see Fig. 18.3). Following this approach, both input spikes into the individual synapses and output spikes generated by the post-synaptic neurons are fast asynchronous digital pulses encoded using the AER protocol. If the activity of the neuron circuits is sparse and their firing rates are biologically plausible (e.g., ranging from a few spikes per second to a few hundred spikes per second), then it is possible to trade-off space with speed very effectively, by time-multiplexing a single (very fast) digital bus to represent many (very slow) neuron axons. The AER input circuits in Fig. 18.2 receive input address-events and decode them to stimulate corresponding columns of synaptic cells. These circuits transmit the events as soon as they arrive and as quickly as possible (e.g., within a few nano-seconds). This frees the shared communication bus to transmit spikes to the cross-bar array, increasing the throughput of the network by use of shared or time-multiplexed communication resources. Integrators and pulse-extenders can be used inside each synapse circuit in the array to convert the fast AER pulses into slower dynamic signals used to emulate synaptic and neural dynamics. On the output side an AER output circuit arbitrates the spikes produced by the neurons and queues output events in case of collisions. The circuit converts the asynchronous spikes produced by the neurons into fast address-events and transmits them on the shared output bus [17].

In both high-density and high-complexity memristive synapse approaches the neurons are typically implemented using Integrate-and-Fire (I&F) models: they receive input currents, that represent the weighted sum of all synaptic contributions, integrate them over time and produce an output spike when their integrated value exceeds the spiking threshold. As a consequence, the timing of the output spike is directly related to the amplitude of the total input current, which in turn depends on the relationship between the timing of the input spike on each synapse and its synaptic weight. In addition to the precise timing of spikes, the neurons can encode signals also with their average firing rates (the average number of spikes produced per second), which are also proportional to their input currents. Therefore the same neuron circuit and

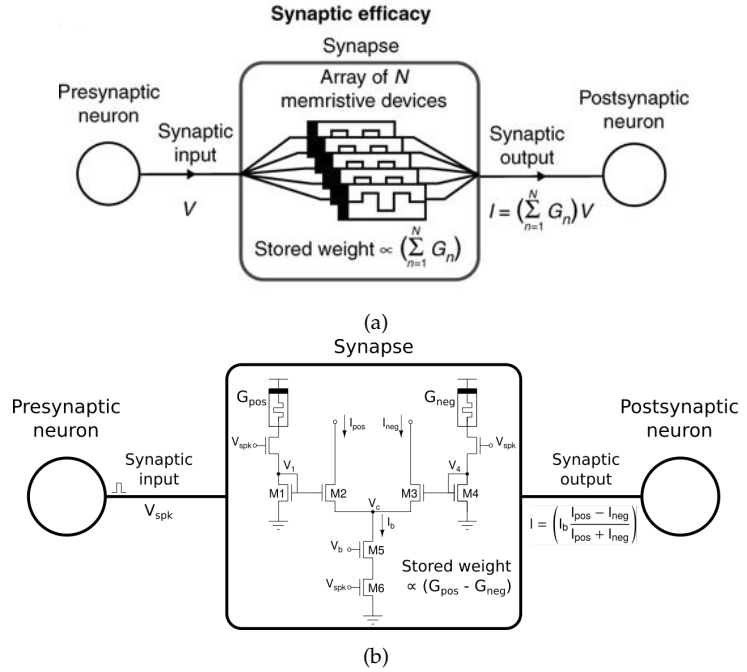

Figure 18.3: Example of compound synapse blocks: (a) Multi-memristive synapse concept, adapted from [24] in which the synaptic weight is represented by the combined conductance of multiple devices; (b) Differential memristive synapse circuit, adapted from [25], in which voltage pulses are directly converted to currents via current-mode normalizer circuits.

architecture can be used to carry out computation on the precise timing of the input/output network signals, or on analog variables encoded in the average firing rates of inputs and outputs. Indeed, it is even possible to use both signal representations (single spike timing and average firing rate) together to carry out complex computations.

18.3 Spike-based learning mechanisms for hybrid memristive-CMOS neuromorphic synapses

Synaptic plasticity plays a crucial role in allowing neural networks to learn and adapt to various input environments. Neuromorphic electronic systems that seek to emulate these learning abilities struggle to meet the seemingly incompatible requirements of reproducing complex plasticity mechanisms in each neuromorphic synapse circuit, while keeping the size of the plastic synapse circuit small, in order to integrate large numbers of synapses per neuron. While many attempts have been made in this respect with pure CMOS mixed signal analog/digital circuits [2, 6, 26–28], several challenges related to the circuit size and the volatility of the learned synaptic weights are still open. In this respect, memristive devices can play a key role in the design of mixed memristive-CMOS learning mechanisms.

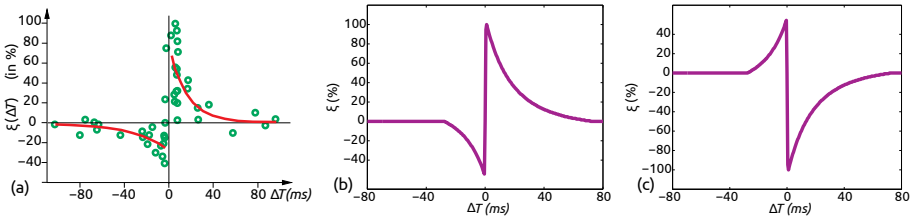

Figure 18.4: STDP learning rule. (a) Experimentally measured STDP function $\xi(\Delta T)$ on biological synapses (data from Bi and Poo [29]), (b) Ideal STDP update function used in computational models of STDP synaptic learning. (c) Anti-STDP learning function for inhibitory STDP synapses.

Here we review different spike-based plasticity models that lend themselves well to hardware implementation using mixed CMOS-memristive circuits. The learning mechanisms and their corresponding circuit design solutions have different properties that trade-off the complexity of the plasticity model emulated with the size and complexity of the circuit proposed.

Spike-timing dependent plasticity (STDP) mechanism

Spike Timing Dependent Plasticity (STDP) is the ability of natural or artificial synapses to change their strength according to the precise timing of individual pre- and/or post-synaptic spikes [29–32]. A comprehensive overview of STDP and its history can be found in [33]. STDP learning in biology is inherently asynchronous and *on-line*, meaning that synaptic incremental update occurs while neurons and synapses transmit spikes and perform computations. This contrasts to more traditional learning rules, like back-propagation [34, 35], where first neurons and synapses perform signal aggregation and neural state update (we call this here "inference phase") and then error signals are computed and corresponding weight updates are applied (we call this here "weight update phase") alternating these two phases during training.

Even early proposals for memristor-based STDP learning implementations used artificial time-multiplexing to alternate continuously and synchronously between "inference" and "weight update" phases, thus requiring global system-wide synchronization. This can become a severe handicap when scaling up systems to arbitrary size. However, it is possible to do a fully asynchronous implementation of memristor-based STDP where "inference" and "weight update" phases happen simultaneously in a natural manner, as in biology [36], where there is no need for any global synchronization.

Figure 18.4(a) shows the change of synaptic strength (in percent) measured experimentally from biological synapses as function of relative timing $\Delta T = t_{pos} - t_{pre}$ between the arrival time t_{pre} of a pre-synaptic spike and the time t_{pos} of generation of a post-synaptic spike. Although the data shows stochasticity, we can infer an underlying interpolated function $\xi(\Delta T)$ as shown in Fig. 18.4(b)

$$\xi(\Delta T) = \begin{cases} a^+ e^{-\Delta T/\tau^+} & if \quad \Delta T > 0 \\ -a^- e^{\Delta T/\tau^-} & if \quad \Delta T < 0 \end{cases} \tag{18.1}$$

For a causal pre to post spike timing relation ($\Delta T > 0$) the strength of the

synapse is increased, while for an anti-causal relation ($\Delta T < 0$) it is decreased. In the case of synapses with negative synaptic strength (as in some artificial realizations), the reversed version shown in Fig. 18.4(c) can be used. Pure CMOS-based VLSI circuit implementations of STDP rules that follow the description of eq. (18.1) have been reported [37, 38], which result in more than about 30 transistors per plastic synapse, thus demonstrating the very high cost of their hardware realization. However, combining memristive crossbars (as those shown in Fig. 18.1) with neurons that comprise learning circuits which send spikes both forward and backwards, it is possible to realize the STDP learning rule in a fully asynchronous manner [39]. Furthermore, by changing the analog shape of the forward and backward voltage pulses produced by the neuron learning circuits, it is possible to massage the STDP learning function. The main drawback of this approach is that the crossbar lines need to be kept "active" for sufficiently long times to guarantee proper overlap between pre- and post-synaptic spikes (sometimes even for up to many milliseconds). This implies that crossbar lines are kept busy for long times, and also that important power is dissipated by the resistive memristors while the spikes are applied. Moreover, such design requires the design of amplifiers which can drive resistive loads. Hence, at least a two-stage amplifier is needed with a high current drive to support such load. However, a problem arises since the resistive load is changing in the process of learning: the compensation required to stabilize the amplifier should be designed to adapt to the changing load. Such design is not only complicated but also is power hungry and consumes a large amount of area on silicon. These issues, however, can be solved by engineering second order memristors having internal dynamics, on which STDP can be induced without forcing pre- and post-synaptic spikes to overlap in time [40]. Under these circumstances, spikes can be made fast in time and resistive power dissipation can be reduced to a minimum.

Spike-timing and -rate dependent plasticity (STRDP) mechanism

The spike-timing and -rate dependent plasticity (STRDP) model was proposed in [41], and is based on work originally presented in [42]. In this model synapses have two stable states on long-time scales (the potentiated and depressed states), but multiple transient states, on short-timescales, that enable a gradual transition between the two stable states. The synaptic weight $w(t)$ is expressed as a function of the synapse internal state variable $X(t)$. Examples of such function can be equivalence ($w(t) = X(t)$) or binary-threshold: $w(t) = 1 if X(t) > w_{th}$ and 0 otherwise, where w_{th} is an arbitrary threshold value. The internal state variable $X(t)$ is updated upon the arrival of a pre-synaptic spike, at time t_{pre}. The direction of the weight update (increase or decrease) depends on the value of the post-synaptic neuron membrane voltage $V_{mem}(t_{pre})$ (whether it is above or below a set threshold θ_V). An additional third factor, the variable $C(t)$, is used to model the intra-cellular Calcium concentration and to determine whether to actually do the weight update or not. Specifically, upon the arrival of the pre-synaptic spike at time t_{pre}, the synaptic weight is updated according to the following equations:

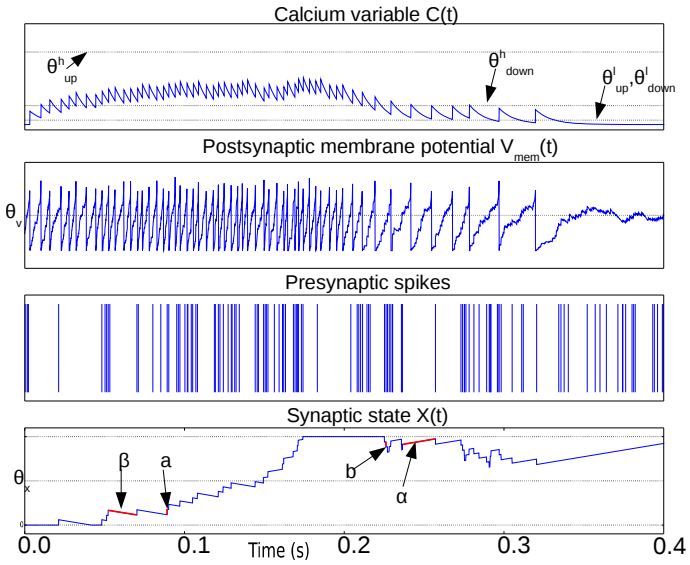

Figure 18.5: Illustration waveforms of the STDRP learning rule showing key parameters from Eqs. 18.2- 18.6 (adapted from [43]).

$$X \rightarrow X + a \quad if \quad V_{mem}(t_{pre}) > \theta_V \quad and \quad \theta^l_{up} < C(t_{pre}) < \theta^h_{up} \tag{18.2}$$

$$X \rightarrow X - b \quad if \quad V_{mem}(t_{pre}) \leq \theta_V \quad and \quad \theta^l_{down} < C(t_{pre}) < \theta^h_{down} \tag{18.3}$$

where a and b are jump sizes, θ_V is a voltage threshold, and θ^l_{up}, θ^h_{up}, θ^l_{down}, and θ^h_{down} are thresholds on the calcium variable. In other words, $X(t)$ is increased if $V_{mem}(t)$ is elevated (above θ_V) when the pre-synaptic spike arrives and decreased if $V_{mem}(t)$ is lower than θ_V at time t_{pre}, provided that the Calcium variable $C(t)$ is in the correct range. The Calcium variable $C(t)$ is an auxiliary variable that corresponds to a low-pass filtered version of the post-synaptic spikes: $C(t)$ is incremented by J_C (which corresponds to magnitude of spike-triggered calcium influx into the cell) at each post-synaptic spike time t_i, and decays with a time constant τ_C:

$$\frac{dC(t)}{dt} = -\frac{1}{\tau_C}C(t) + J_C \sum_i \delta(t - t_i) \tag{18.4}$$

The dependence of the weight updates on $C(t)$ allows the learning rule to enable/disable the weight updates based on the long-term average of post-synaptic activity. This implements a "stop-learning" condition that allows the network to stop changing weights when it is performing correctly, and allows the system to keep the learning process always enabled, without having to separate a "training phase" from a "test phase" as is typically done in conventional artificial neural network applications.

In parallel to the spike-driven weight updates described above, $X(t)$ is continuously and slowly driven towards one of two stable values, depending on whether it is above or below an additional threshold parameter θ_X:

$$\frac{dX}{dt} = \alpha \qquad if \qquad X > \theta_X \tag{18.5}$$

$$\frac{dX}{dt} = -\beta \qquad if \qquad X \leq \theta_X \tag{18.6}$$

The state variable $X(t)$ is bounded above and below by the two stable states X_{high} and X_{low} which are not shown in the equations to simplify the notation. Figure 18.5 illustrates the relevant wave-forms and parameters of the spike-based voltage dependent learning rule.

Although this learning rule has been shown to reproduce, on average, the classical STDP phenomenology [41], it differs from the vast majority of STDP rules in that it does not explicitly depend on the precise timing of both pre- and post-synaptic neuron spikes. The compatibility with the classical STDP learning rule comes about through the rule's dependence on the post-synaptic neuron's membrane potential: a pre-synaptic spike that occurs when the post-synaptic membrane potential is high will potentiate the synapse and will likely produce a post-synaptic spike shortly after. Thus, the synapse tends to get potentiated in pre-before-post scenarios. The synapse also tends to get depressed in post-before-pre scenarios because the membrane potential is usually low for some time after a post-synaptic spike is emitted, and a pre-synaptic spike arriving in this interval will depress the synapse. However, as this rule also has access to post-synaptic neuron's rate information, through the $C(t)$ signal, it can reproduce effects beyond classical pair-wise STDP, such as increased potentiation at high post-synaptic firing rates and increased depression at low post-synaptic firing rates [44].

Spike-based stochastic weight update rules

Filamentary memristive devices have large variations in their operational parameters which stem from the underlying switching mechanisms that are based on filament formation. This switching mechanism exhibits stochastic behavior due to the thermally activated filament formation process [22, 45–47]. Filament formation in memristive devices is typically bias-dependent and can be explained by the hopping of positively charged particles in a thermally activated process [22]. The hopping rate is exponentially related to the activation energy and linearly dependent in time:

$$\Gamma = 1/\tau = \upsilon e^{-E_a(V)/k_B T} \tag{18.7}$$

where υ is the attempt frequency for particle hopping, k_B is the Boltzmann constant and T is the absolute temperature. The bias dependent nature of the switching characteristics results in a stochastic process which follows a Poisson distribution. The Poisson distribution implies that the switching events are independent from one another. Therefore the probability of a switching event occurring within Δt at time t is:

$$P(t) = \frac{\Delta t}{\tau} e^{-t/\tau} \tag{18.8}$$

where τ is the characteristic wait time and is an exponential function of the voltage applied across the device:

$$\tau(V) = \tau_0 e^{-V/V_0} \tag{18.9}$$

The parameters τ_0 and V_0 are fitting parameters that depend on the physical characteristics of the memristive device and can be found by experimental measurements (e.g., see [22]). This intrinsic probabilistic property of memristive devices can be exploited for implementing stochastic learning in neuromorphic architectures, to overcome the limitations and problems typically found with non-linear conductance changes in deterministic learning approaches [48] and to reduce the network sensitivity to their variability [49].

The learning algorithm that we use for achieving such features is based on the *delta rule* [50]. This rule minimizes the cost function of a single-layer neural network defined as the error between a desired target value T and the variable y calculated as weighted sum of the network input:

$$y = \sum_i (w_i x_i) \tag{18.10}$$

where w_i and x_i are the ith synaptic weight and input respectively. In the original formulation [50] the network inputs x_i were binary signals and the output was a thresholded and binarized function of the weighted sum variable. In neuromorphic architectures the x_i inputs are spiking events and the network output corresponds to the activity of the spiking neurons. If one considers the average firing rate of the neurons, then the output is a sigmoidal function of y. According to this rule, the weight change of the ith input synapse should be proportional to: $\Delta W_{ji} =\propto (T - y)x_i$ [51]. If we consider binary synapses with synaptic weights represented by set or reset memristive devices, the analog nature of the weight change can be interpreted as the probability of switching rather than a gradual change in the device conductance. Taking into account the stochastic filament formation behavior described above, the synapse probability of switching for $t << \tau$ can be written as

$$P(t) = \frac{\Delta t}{\tau} = \frac{\Delta t}{\tau_0 e^{V/V_0}} \tag{18.11}$$

To map this stochastic property to the delta rule the voltage across the device should be set to:

$$V/V_0 = \ln(T - y) \tag{18.12}$$

Therefore, with this setting, upon the arrival of an input spike event x_i, the probability of switching thus becomes:

$$P(t) = \Delta t e^{\ln(T_j - y_j)} x_i = \Delta t (T - y) x_i \tag{18.13}$$

In this the requirement to linearly change synaptic weights with gradual changes in memristive device conductances has been converted to switching binary devices with a probability that is linearly proportional to the error.

The circuits required to implement this stochastic rule on a neuromorphic chip have been recently proposed in [49]. They comprise:

- a block that compares the sum of the synaptic currents (obtained by simply using Kirchhoff's current law on the neuron's input node) with a target current provided by the system
- a circuit that produces a a ramping voltage whose slope is proportional to the logarithmic value of the error, as specified by eq. (18.12) Such ramp voltage has been shown to be able to modulate the probabilty of the switching of the device depending on the final value it reaches by the end of a programming time [52].

Behavioral simulations of such probabilistic mechanism [49] have shown that this stochastic learning rule provides promising results for example in classical benchmark tasks, such as the classification of handwritten MNIST characters [53], and that performance improves when combining memristive single devices into compound multi-memristive devices, as explained in Section 18.2.

Recent results on STDP learning with binary weights following stochastic updates with additional regularizations, such as homeostasis and moving and double thresholds [54], present an excellent potential to be exploited on memristors restricted to binary values.

Comparison between the spike-based learning architectures

STDP and STRDP are spike-based hebbian rules which at a network level could be utilized for example in Hopfield networks, Restricted Boltzmann machines and Deep belief networks where they minimize the energy function of such networks. Moreover, they are used as unsupervised learning paradigm to cluster the data in competitive learning structures in which the distance between the moving average of the cluster and each data point is minimized [55]. However, to learn a specific target function in the hardware, it is much more desirable to define a more specific cost function and employ gradient descent for its optimization. Deriving an update rule based on this optimization algorithm results in Delta rule for a one layer network and extends to "back-propagation" using chain rule for deeper networks. Back-propagation update rule however, does not pass the locality criteria required for being implemented in hardware. However, recently it has been shown that such update rule can be implemented in a local fashion to approximate back propagation which makes such an algorithm even more powerful [56, 57].

Given the 7 bits of precision reported in [58], this learning rule can take advantage of the "almost" analog nature of the memory by translating the error of the network to the number of pulses applied to the device. Such circuit is presented in [59] in which the network error modulates the frequency of a ring oscillator and hence the device can be tuned more precisely to a desire value.

18.4 Spike-based implementation of the neuronal intrinsic plasticity

Neurons have multiple parameters which elicits certain computational features. Such properties include but are not limited to its time constant, refractory period and adaptation time constants. These parameters are determined by the conductance of the nerurons' membrane proteins. Biological neurons

continuously adapt these conductances to maximize information transfer and minimize power consumption [60]. This local adaptive mechanism is described as neuronal intrinsic plasticity (IP) which is shown to act as a neuronal homeostatic plasticity regulating the network's activity [61]. These neuron parameters are in the range of biological time constants in the order of milliseconds which are emulated in hardware using sub-threshold circuits [5] biased using a central bias generator [62].

Currently, the state of the art NPs compromise variety and share these parameters for all the neurons and synapses on a single core (thousands of neurons and synapses). If parameters were not shared the static power consumption and the area consumed by wires (metal lines) running across the chip for connecting the biases grows linearly with the number of parameters.

As memristive devices are adaptable conductances, it is a natural implementation to use them as the replacement for the biases and use local plasticity rules to adapt them based on the neuron's activity. However, given the millisecond range of time constant required, the memristive devices should always be in their High Resistive State (HRS). Measurements in [52] shows that the mean HRS value of memristive devices in a 4kb array is an exponential function of the reset voltage applied across the device which follows a lognormal distribution. The circuit introduced in section 18.3 could be used to apply the appropriate reset voltage acorss the device until the neuron's firing rate falls into the target range of activity. In such hybrid systems each circuit model has its own parameters set by the incorporated RRAM without area or static power overhead which also enables the self-organisation of individual parameters locally.

18.5 Scalable mixed memristive-CMOS multi-core neuromorphic computing systems

To create complete neuromorphic computing systems that can support large-scale networks, including deep neural network and convolutional neural network architectures, it is necessary to develop an infrastructure to integrate in a single VLSI die multiple spiking neural network cores and to connect the neurons and synapses of each core among all other neurons and synapses. Examples of pure CMOS neuromorphic computing systems that comprise auxiliary circuits for interconnecting multiple cores and creating large-scale neural processing systems have already been developed in the past [1–4, 7]. These all make use of the AER communication protocol to route spikes from source neurons to destination synapses, and implement different types of routing schemes. One of the most critical factors for building routing schemes that can support very large scale networks is the memory required to store the network connectivity parameters (routing tables and weights). While some solutions have resorted to using external DRAM memory chips [1], others chose to use on-chip SRAM circuits [2, 3, 7]. Both solutions have advantages and disadvantages: external DRAM chips can store very large amounts of memory, but the energy required to transfer the data from the memory chip to the neuromorphic processor is substantially larger than that used by on-chip SRAM circuits; on the other hand SRAM cells require at least 6 transistors per bit, and therefore occupy a substantial amount of area on the silicon die. The use of memristive devices can have a significant impact on the design of future

neuromorphic processor chips also if used as classical digital memory circuits to store the routing data. They promise to significantly increase the density and at the same time reduce the power consumption, as the wr"Siting of the data would be performed only at the onset of the experiment, during the definition of the network connectivity tables. Furthermore, when stored in DRAM or SRAM cells, the network parameters would be deleted when the chips are reset, and when power is removed. This would require users to upload all the parameters again when the system is powered-up. This can be particularly problematic in large networks, whereby the storing and initialization of all the system parameters can take a significant amount of time. By virtue of their non-volatility features [63], these memristive devices will save also the configuration/start-up time, when booting up the system.

18.6 Conclusions and discussion

In this Chapter we have given a quick overview of some techniques to exploit memristive devices for implementing neuromorphic computing and learning systems. In such systems, most of chip area is devoted to the implementation of synapses. Therefore, it is in these components where the use of memristors can provide a significant boost for system density. We have quickly discussed two main approaches. One, in which a synapse is built using one single memristor, giving the highest possible density, but at the cost of introducing significant overheads for the peripheral circuits and introducing timing and power constraints (unless second order memristors are used, which is not discussed in this Chapter). And a second one, in which synapses are made more complex by combining memristors and CMOS transistors, resulting in less complex and more efficient peripheral circuits. Several STDP learning mechanisms exploiting memristors are discussed, ranging from plain STDP, STRDP, to stochastic STDP. Finally, some quick scalability considerations are discussed.

Overall, it is clear that memristive devices have a great potential for providing new ways of performing neuromorphic computing, including learning, with highly compact as well as low power physical realizations. On the other hand it is also true that by today such realizations remain at the lab level, at small scale, and practical issues remain to be solved. For example, at the time of this writing, we are only aware of one hybrid CMOS-memristor technology being available for circuit design researchers as MPW (multi project wafers) through Europractice (www.europractice-ic.com) or CMP (mycmp.fr) that offers monolithic 1T1R (one memristor with one NMOS selector transistor) devices that can store 1-bit. Still these memristors require one selecting transistor in series each, thus increasing significantly its size, while occupying Si area, as opposed to pure memristors which could be added on top of the CMOS transistor layer without consuming transistor space. These selectors are necessary to limit the current during forming and writing operations. However, researchers are investigating on non-transistor-based selectors that could share the same footprint than the memristors [64]. This, together with the successful use of second-order memristors for learning could potentially yield to compact and low power self-learning neuromorphic systems. Another not yet fully solved problem is the issue of using the memristors as reliable analog memory. Some preliminary

results showing up to 7-bit equivalent analog memory for a single $10\mu m \times 10\mu m$ memristor have been reported recently [58]. Exploiting the analog memory potential of memristors reliably, would be an additional important twist to achieve ultra high density memories co-existing with CMOS transistors for ultra-dense low-power massive parallel computation (wether neuromorphic or not). For example, imagine that an equivalent of 8-bit per memristor could be achieved while assembling them at a pitch of 20nm within a layer, and that one could stack 10 of such layers, over a surface of a $1cm^2$ chip. This would be equivalent to digital non-volatile memory of 2.5TB, working very tightly with CMOS computing fabric, faster than an L1 cache memory does today.

Bibliography

[1] S. Furber, F. Galluppi, S. Temple, and L. Plana, "The SpiNNaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

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

[3] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014, ISSN: 0036-8075, 1095-9203.

[4] J. Schemmel, D. Bruderle, A. Grubl, *et al.*, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, IEEE, 2010, pp. 1947–1950.

[5] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1367–1388, Sep. 2014, ISSN: 0018-9219.

[6] N. Qiao, H. Mostafa, F. Corradi, *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in Neuroscience*, vol. 9, no. 141, pp. 1–17, 2015.

[7] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[8] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, p. 384 010, 2013.

[9] M. Payvand, M. Nair, L. Müller, and G. Indiveri, "A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: From mitigation to exploitation," *Faraday Discussions*, pp. 1–13, 2018.

[10] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.

[11] E. Neftci, J. Binas, U. Rutishauser, *et al.*, "Synthesizing cognition in neuromorphic electronic systems," *Proceedings of the National Academy of Sciences*, vol. 110, no. 37, E3468–E3476, 2013.

[12] J. Backus, "Can programming be liberated from the von neumann style?: A functional style and its algebra of programs," *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, 1978.

[13] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, 2015.

[14] S. Habenschuss, Z. Jonke, and W. Maass, "Stochastic computations in cortical microcircuit models," *PLoS computational biology*, vol. 9, no. 11, e1003311, 2013.

[15] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 860–880, May 2014, ISSN: 0018-9219.

[16] S. Deiss, R. Douglas, and A. Whatley, "A pulse-coded communications infrastructure for neuromorphic systems," in *Pulsed Neural Networks*, W. Maass and C. Bishop, Eds., MIT Press, 1998, ch. 6, pp. 157–78.

[17] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address-events," *IEEE Transactions on Circuits and Systems II*, vol. 47, no. 5, pp. 416–34, 2000.

[18] P. Merolla, J. Arthur, B. Shi, and K. Boahen, "Expandable networks for neuromorphic chips," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 2, pp. 301–311, Feb. 2007.

[19] E. Chicca, A. Whatley, P. Lichtsteiner, *et al.*, "A multi-chip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity," *IEEE Transactions on Circuits and Systems I*, vol. 5, no. 54, pp. 981–993, 2007.

[20] P. Dayan and L. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA, USA: MIT Press, 2001, ISBN: 9780262541855.

[21] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, *et al.*, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[22] S. H. Jo, T. Chang, I. Ebong, *et al.*, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[23] M. Zidan, H. Omran, R. Naous, *et al.*, "Single-readout high-density memristor crossbar," *Scientific reports*, vol. 6, p. 18 863, 2016.

[24] I. Boybat, M. L. Gallo, T. Moraitis, *et al.*, "Neuromorphic computing with multi-memristive synapses," *Nature communications*, vol. 9, p. 2514, 2018.

[25] M. V. Nair and G. Indiveri, *A differential memristive current-mode circuit*, European patent application EP 17183461.7, Filed 27.07.2017, Jul. 2017.

[26] S. Nease and E. Chicca, "Floating-gate-based intrinsic plasticity with low-voltage rate control," in *International Symposium on Circuits and Systems ISCAS 2016*, IEEE, 2016, pp. 2507–2510.

[27] F. L. M. Huayaney, S. Nease, and E. Chicca, "Learning in silicon beyond STDP: A neuromorphic implementation of multi-factor synaptic plasticity with calcium-based dynamics," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2189–2199, 2016.

[28] S. Schmitt, J. Klähn, G. Bellec, *et al.*, "Neuromorphic hardware in the loop: Training a deep spiking network on the BrainScaleS wafer-scale system," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 2227–2234.

[29] G.-Q. Bi and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.

[30] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 6595, p. 76, 1996.

[31] T. Masquelier, R. Guyonneau, and S. Thorpe, "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains," *PLoS ONE*, vol. 3, no. 1, e1377, Jan. 2008.

[32] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs," *Science*, vol. 275, pp. 213–215, 1997.

[33] H. Markram, W. Gerstner, and P. Sjöström, "Spike-timing-dependent plasticity: A comprehensive overview," *Frontiers in Synaptic Neuroscience*, vol. 4, no. 2, pp. 1–3, 2012.

[34] P. Werbos, *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. Wiley. com, 1994, vol. 1.

[35] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[36] S. Saighi, C. Mayr, B. Linares-Barranco, *et al.*, "Plasticity in memristive devices," *Frontiers in Neuroscience*, vol. 9, no. 51, 2015.

[37] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike–timing dependent plasticity," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 211–221, Jan. 2006.

[38] M. R. Azghadi, N. Iannella, S. Al-Sarawi, G. Indiveri, and D. Abbott, "Spike-based synaptic plasticity in silicon: Design, implementation, application, and challenges," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 717–737, May 2014, ISSN: 0018-9219.

[39] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems," *Frontiers in Neuroscience*, vol. 7, no. 2, 2013, ISSN: 1662-453X.

[40] S. Kim, C. Du, P. Sheridan, *et al.*, "Experimental demonstration of a second-order memristor and its ability to biorealistically implement synaptic plasticity," *Nano letters*, vol. 15, no. 3, pp. 2203–2211, 2015.

[41] 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.

[42] S. Fusi, M. Annunziato, D. Badoni, A. Salamon, and D. Amit, "Spike–driven synaptic plasticity: Theory, simulation, VLSI implementation," *Neural Computation*, vol. 12, pp. 2227–58, 2000.

[43] H. Mostafa, A. Khiat, A. Serb, *et al.*, "Implementation of a spike-based perceptron learning rule using TiO2-x memristors," *Frontiers in Neuroscience*, vol. 9, no. 357, 2015.

[44] P. Sjöström, G. Turrigiano, and S. Nelson, "Rate, Timing, and Cooperativity Jointly Determine Cortical Synaptic Plasticity," *Neuron*, vol. 32, no. 6, pp. 1149–1164, Dec. 2001.

[45] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, no. 13, pp. 5872–5878, 2013.

[46] S. Ambrogio, S. Balatti, V. Milo, *et al.*, "Neuromorphic learning and recognition with one-transistor-one-resistor synapses and bistable metal oxide RRAM," *IEEE Transactions on Electron Devices*, vol. 63, no. 4, pp. 1508–1515, 2016.

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

[48] J. Woo, K. Moon, J. Song, *et al.*, "Improved synaptic behavior under identical pulses using AlOx/HfO2 bilayer RRAM array for neuromorphic systems," *IEEE Electron Device Letters*, vol. 37, no. 8, pp. 994–997, 2016.

[49] M. Payvand, L. K. Muller, and G. Indiveri, "Event-based circuits for controlling stochastic learning with memristive devices in neuromorphic architectures," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*, IEEE, 2018, pp. 1–5.

[50] B. Widrow and M. Hoff, "Adaptive Switching Circuits," in *1960 IRE WESCON Convention Record, Part 4*, New York: IRE, 1960, pp. 96–104.

[51] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley, 1991.

[52] T. Dalgaty, M. Payvand, B. De Salvo, *et al.*, "Hybrid cmos-rram neurons with intrinsic plasticit," in *International Symposium on Circuits and Systems (ISCAS), 2019*, IEEE, 2019.

[53] *The MNIST database of handwritten digits*, Yann LeCun's web-site, May 2012.

[54] A. Yousefzadeh, E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "On practical issues for stochastic stdp hardware with 1-bit synaptic weights," *Frontiers in neuroscience*, vol. 12, 2018.

[55] R. Kreiser, T. Moraitis, Y. Sandamirskaya, and G. Indiveri, "On-chip unsupervised learning in winner-take-all networks of spiking neurons," in *Biomedical Circuits and Systems Conference, (BioCAS), 2017*, IEEE, Oct. 2017, pp. 424–427.

[56] J. Sacramento, R. P. Costa, Y. Bengio, and W. Senn, "Dendritic error backpropagation in deep cortical microcircuits," *arXiv preprint arXiv:1801.00062*, 2017.

[57] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers in Neuroscience*, vol. 11, p. 324, 2017, ISSN: 1662-453X.

[58] S. Stathopoulos, A. Khiat, M. Trapatseli, *et al.*, "Multibit memory operation of metal-oxide bi-layer memristors," *Scientific reports*, vol. 7, no. 1, p. 17 532, 2017.

[59] M. Payvand and G. Indiveri, "Spike-based plasticity circuits for always-on on-line learning in neuromorphic systems," in *International Symposium on Circuits and Systems (ISCAS), 2019*, IEEE, 2019.

[60] M. Stemmler and C. Koch, "How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate," *Nature Neuroscience*, vol. 2, pp. 521–527, 1999.

[61] J. Triesch, "Synergies between intrinsic and synaptic plasticity mechanisms," *Neural Computation*, vol. 19, pp. 885–909, 2007.

[62] T. Delbruck, R. Berner, P. Lichtsteiner, and C. Dualibe, "32-bit configurable bias current generator with sub-off-current capability," in *International Symposium on Circuits and Systems, (ISCAS), 2010*, IEEE, Paris, France: IEEE, 2010, pp. 1647–1650.

[63] D. Ielmini and R. Waser, *Resistive Switching: From Fundamentals of Nanoionic Redox Processes to Memristive Device Applications*. John Wiley & Sons, 2015.

[64] Z. Wang, S. Joshi, S. E. Savel'ev, *et al.*, "Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing," *Nature materials*, vol. 16, no. 1, p. 101, 2017.