


Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions

Arindam Basu , *Member, IEEE*, Jyotibdha Acharya, *Member, IEEE*, Tanay Karnik, *Fellow, IEEE*, Huichu Liu, *Member, IEEE*, Hai Li, *Senior Member, IEEE*, Jae-Sun Seo, *Senior Member, IEEE*, and Chang Song

Abstract—In this paper, we present a survey of recent works in developing neuromorphic or neuro-inspired hardware systems. In particular, we focus on those systems which can either learn from data in an unsupervised or online supervised manner. We present algorithms and architectures developed specially to support on-chip learning. Emphasis is placed on hardware friendly modifications of standard algorithms, such as backpropagation, as well as novel algorithms, such as structural plasticity, developed specially for low-resolution synapses. We cover works related to both spike-based and more traditional non-spike-based algorithms. This is followed by developments in novel devices, such as floating-gate MOS, memristors, and spintronic devices. CMOS circuit innovations for on-chip learning and CMOS interface circuits for post-CMOS devices, such as memristors, are presented. Common architectures, such as crossbar or island style arrays, are discussed, along with their relative merits and demerits. Finally, we present some possible applications of neuromorphic hardware, such as brain-machine interfaces, robotics, etc., and identify future research trends in the field.

Index Terms—Neuromorphics, machine learning, learning systems, adaptive systems, low-power electronics, artificial neural networks, neural network hardware, MOS integrated circuits.

LIST OF ABBREVIATIONS

ADALINE	Adaptive Linear Combiner,
AER	address event representation,
ANN	artificial neural network,
ARBP	adaptive random backpropagation,
ASRBP	adaptive skipped random backpropagation,
BMI	Brain Machine Interface,
CAB	computational analog blocks,
CNN	convolutional neural network,
DNN	deep neural network,
DRAM	Dynamic random-access memory,
D-STDP	doublet spike time dependent plasticity,
DW	domain-wall,
DWM	domain wall magnet,

ELM	extreme learning machine,
eNVM	embedded nonvolatile memory,
EPSC	excitatory postsynaptic current,
eRBP	event-driven RBP,
FeFET	ferroelectric field-effect-transistor,
FGMOS	floating-gate MOS,
FL	free layer,
FM/HM	ferromagnet/heavy metal,
FPAA	field programmable analog array,
FGPA	field programmable gate arrays,
HEI	hot electron injection,
IoT	Internet of Things,
LIF	Leaky-integrate-fire,
LMS	Least Mean Square,
LSV	lateral spin valve,
LTD	long-term depression,
LTP	long-term potentiation,
MAC	multiply and accumulate,
ME	magnetoelectric,
MEA	microelectrode array,
MTJ	Magnetic tunnel junctions,
NCS	neuromorphic computing systems,
NVM	nonvolatile memory,
PARCA	parallel architecture with resistive crosspoint array,
PCM	phase change memory,
PL	pinned layer,
RBP	random backpropagation,
RRAM	resistive random-access memory,
RUSD	randomized unregulated step descent,
RVFL	Random vector functional link,
SBF	single-bit failure,
SDSP	Spike Driven Synaptic Plasticity,
SHE	spin-hall effect,
SNN	spiking neural network,
SOT	spin-orbital torque,
SOUL	Sign based Online Update Learning,
SP	structural plasticity,
SRAM	Static random-access memory,
SRBP	skipped random backpropagation,
STDP	spike time dependent plasticity,
STT-RAM	spin transfer torque random-access memory,
T-STDP	triplet spike time dependent plasticity

Manuscript received November 16, 2017; revised February 21, 2018; accepted March 8, 2018. Date of publication March 15, 2018; date of current version April 3, 2018. This paper was recommended by Guest Editor Y. K. Chen. (*Corresponding author: Arindam Basu.*)

A. Basu and J. Acharya are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: arindam.basu@ntu.edu.sg).

T. Karnik is with the Microarchitecture Research Lab, Intel Corporation, Hillsboro, OR 97124 USA.

H. Liu is with the Microarchitecture Research Lab, Intel Corporation, Santa Clara, CA 95054 USA.

H. Li and C. Song are with Duke University, Durham, NC 27519 USA.

J.-S. Seo is with Arizona State University, Tempe, AZ 85287 USA.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/JETCAS.2018.2816339

I. INTRODUCTION

THE recent success of “deep neural networks” (DNN) has renewed interest in machine learning and, in particular, bio-inspired machine learning algorithms. DNN refers to neural networks with multiple layers (typically two or more) where the neurons are interconnected using tunable weights.

Although these architectures are not new, availability of massive amount of data, huge computing power and new training techniques (such as unsupervised initialization, use of rectified linear units as the neuronal nonlinearity, regularization using dropout or sparsity, etc. [1], [2]) to prevent the networks from over-fitting have led to its great success in recent times. DNN has been applied to a variety of fields such as image classification [3], [4], face recognition in images [5], word recognition in speech [6], [7], natural language processing [1], [8], game playing [9], and the success stories of DNN continue to increase every day.

However, the common training method in deep learning, such as back propagation, tunes the weights of neural networks based on the gradient of the error function, which requires a known output value for every input. It would be difficult to use such supervised learning methods to train and adapt to real-time sensory input data that are mostly unlabeled. Furthermore, training and classification phases of deep neural networks are typically separated, such that training occurs in the cloud or high-end graphics processing units, while their weights or synapses are fixed during deployment for classification. However, this makes it difficult for the neural network to continuously adapt to input or environment changes in real-world applications. By adopting unsupervised and semi-supervised learning rules found in biological nervous systems, we anticipate to enable adaptive systems for many real-time applications with a large amount of unlabeled data, similar to how humans analyze and associate sensory input data. In this review, online supervised learning schemes are also considered to be within the purview of adaptive algorithms. Energy-efficient hardware implementation of these adaptive learning systems is particularly challenging due to intensive computation, memory, and communication that are necessary for online, real-time learning and classification. “Neuromorphic” engineering is a possible solution to this energy efficiency problem as well. Coined by Carver Mead in his seminal paper [10], the term neuromorphic engineering was used to imply that using analog or physical computing substrates similar to biology could yield orders of magnitude energy reduction for applications processing noisy sensory data. The term is now often used in a broader sense to imply neuro-inspired techniques, be it hardware or algorithms. Apart from analog computing, some commonly observed features in neuromorphic hardware systems are typically distributed memory as opposed to von-Neumann models and spike or pulse based computing.

Neuromorphic engineering was recently voted as one of the top ten emerging technologies by the World Economic Forum [11] and the market for neuromorphic hardware is expected to grow to $\sim \$1.8B$ by 2023/2025 [12], [13]. With the massive growth in big-data and internet of things (IoT), the requirement for such hardware is only going to increase. However, cross-layer innovations on algorithms, architectures, circuits, and devices are required to enable adaptive intelligence especially on embedded systems with severe power and area constraints. In this survey paper, we review all aspects of *adaptive* neuromorphic systems across algorithms, devices, circuits, and architectures. The focus of the paper is also clearly shown in Figure 1 to put it in context of general machine learning research. Recent surveys have either focused solely on deep learning hardware for inference [14], CMOS adaptive synapses [15] or on some specific aspects of neuromorphic systems [16], [17]. Another review [18] has attempted

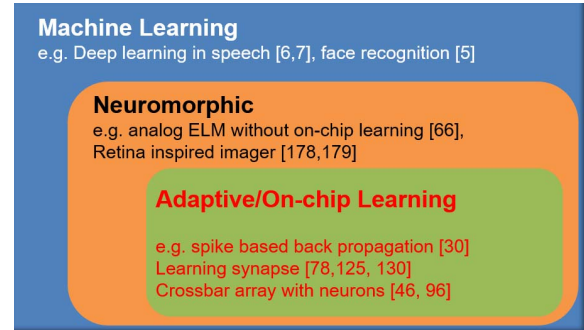


Fig. 1. Neuromorphic systems are a subset of machine learning systems. The focus of this paper is a further subset of that—ones with on-chip learning or adaptation.

to paint a roadmap for future development of neuromorphic systems focussing more on dynamics and role of dendrites in neural computation. In contrast, we hope to give a holistic picture of all the aspects of adaptive neuromorphic hardware systems including recent technology like spin devices as well as machine learning algorithms like random backpropagation etc. in this paper.

In Section II, new learning algorithms that are suitable for hardware implementations are presented. Section III presents new devices with properties amenable to implementing neuromorphic algorithms followed by Section IV on CMOS circuits to implement some of these algorithms or interface with the novel devices presented earlier. High-level architectures for large-scale neuromorphic hardware are presented in Section V, followed by conclusions and future directions in the last section. Given the huge volume of work in this field over the last few years, we have categorized the reviewed papers according to two criteria: (1) spike based versus non-spike based and (2) CMOS implementation versus post-CMOS devices used. We feel that this categorization helps in organizing the work well.

II. LEARNING ALGORITHMS

In this section, we review some of the most popular learning algorithms that have been implemented on neuromorphic hardware. In some cases, modifications to the original algorithm have been proposed to make it feasible to implement on low-precision platforms with only locally available information. Most recent work has focussed on spike based learning rules while only a few have considered non-spike based ones [19], [20]. Hence, we discuss the spike based rules in details while summarizing the non-spike based ones.

Before going into details, let us review some of the desirable properties of the algorithms.

- **Computational complexity** i.e number of operations needed per weight update should be low.
- **Locality** of the weight update rule is desired. In other words, the weight update rule should ideally only need information related to the variables in the computational element (neuron) connected to it.
- **Convergence speed** of the algorithms should be as fast as possible.

A. Least Mean Square (LMS), Perceptron

Two of the oldest learning rules derived for non-spiking networks were the LMS and the Perceptron learning rules [22]. They were typically used to train single layer neural networks. Both rules follow the so-called “Minimum Disturbance

Principle” [22] that aims to correct the error at every step by minimal change in parameters to maximize memory capacity.

1) *LMS*: The simple neural network under consideration here is called the Adaptive Linear Combiner (ADALINE) and is described by the following equations:

$$s = \overline{X}^T \overline{W} \quad (1)$$

$$e = d - s \quad (2)$$

where \overline{X} denotes the input vector, \overline{W} is the weight vector, d is the desired output and e is the corresponding error. In this case, the LMS learning rule prescribes a weight update [23] as given by:

$$\Delta \overline{W} = \alpha \frac{e \overline{X}}{|\overline{X}|^2} \quad (3)$$

where α is typically chosen to be between 0.1 and 1 for a good tradeoff between stability and convergence speed. It can be shown that the error reduces at every step at a fixed rate dependent on α . This rule is known to minimize mean squared error. The LMS algorithm has often been implemented in adaptive circuits especially for filters [24], [25].

2) *Perceptron*: In this case, we consider a non-linear network by adding a threshold function at the output of the ADALINE. The new output $y = \text{sgn}(s)$ where $\text{sgn}()$ is the signum function [22]. The new error is now defined as $e = d - y$. Then, the perceptron weight update [26] is given by:

$$\Delta \overline{W} = \alpha \frac{e}{2} \overline{X} \quad (4)$$

Here, α does not affect stability but only affects rate of convergence. The perceptron rule is guaranteed to converge to classification error of 0 if the patterns presented to it are linearly separable. However, for non-linearly separable patterns, the algorithm does not guarantee low classification errors. Other variants of this rule including margin [27] have also been proposed.

B. Backpropagation and Variants

Backpropagation and gradient descent have been the most popular learning algorithms for training both shallow and deep neural networks for past decades. It is simple, intuitive and works well for most of the real world applications. In standard gradient descent backpropagation, the error signal is backpropagated from output layer to the deeper layers on a neuron-by-neuron basis [28]. Mathematically, the output of layer i of a neural network, \overline{h}_i is given by:

$$\overline{h}_i = f(W_{ij} \overline{h}_j + \overline{b}_i) \quad (5)$$

where W_{ij} are the weights connecting layer j and layer i , \overline{b}_i is the bias and $f(\cdot)$ is the non-linear activation function. The weight update ΔW_{ij} is given by:

$$\Delta W_{ij} = -\eta \frac{\partial \overline{E}}{\partial W_{ij}} \quad (6)$$

where η is the learning rate and \overline{E} is the error at the output layer.

In spite of the huge success of backpropagation in different forms of neural networks, several issues have been pointed out over the years regarding its hardware implementation and biological plausibility. Typical implementation of backpropagation necessitate use of high precision weights and

smooth activation functions (tanh, sigmoid etc.) as well as a significant number of multiply and accumulate (MAC) operations for both forward and backward pass. From a hardware perspective, these features make backpropagation costly, both in terms of memory and power. Moreover, backpropagation requires global information about the errors well as weights in backward pass that are exactly symmetric with respect to the weights in the forward pass. This weight transport problem [29], along with alternating forward and backward passes and precise floating point weights make standard backpropagation an implausible learning technique for brain as well as neuromorphic architectures [30].

A proposed solution to reduce the computational complexity and optimize memory resources is the use of pipelined backpropagation [31] and binary state network [32]. In binary state network, the output of a neuron can be unipolar (0/1) or bipolar ($-1/1$) binary. This modification transforms the MAC operations in the forward pass to simple additions and subtractions. In [31], error ternarization was proposed which removes the requirement of MAC operations even in the backward pass. Thus, all the network operations become equivalent to synaptic operations (SynOps) which can facilitate efficient neuromorphic implementations. In pipelined backpropagation, the network has access to the previous network states and therefore, can perform delayed weight updates during the subsequent forward pass without the need for an explicit backward pass. This architecture reduces the number of memory fetches and lookups (corresponding to backward pass) significantly. The reduction of redundant off-chip memory fetches can also reduce the power overhead appreciably [31].

Another significant constraint regarding hardware implementation of neural networks is the requirement of high precision floating point weights and gradients. This problem is addressed by two approaches: (1) off-chip learning using high resolution weights and gradients and consequently rounding off the weights to lower resolution for on-chip implementation and (2) on-chip learning using low resolution weights. In [33] randomized rounding has been used to map high resolution gradients to low resolution discrete values. Moreover, k-means clustering has been used to map high resolution weights to low resolution values. One of the significant low resolution learning methods for on-chip learning is randomized unregulated step descent (RUSD) introduced in [34]. This method combines unregulated step descent (USD) with randomized rounding i.e. the step size for weight update is large (low resolution) but the weight update probability is low. So, the overall average weight update is small (comparable to high resolution weight update case).

A simple yet elegant solution has been proposed in [28] to solve the weight transport and global information problem of the backpropagation algorithm. In the proposed random backpropagation (RBP), the errors are backpropagated through a random weight matrix(B) instead of the transpose of forward pass weights (W^T). So, the error update formula changes from $\Delta W \propto -W^T e x^T$ to $\Delta W \propto B e x^T$. Therefore, RBP is spatially and temporally local [30] and absence of feedback weight constraints makes it possible for a feedback architecture consisting of a separate feedback channel which is biologically more plausible. A possible explanation as to why this architecture works, is that the network weights adjust themselves to the random feedback weights and learns afterwards. The RBP has also shown promise in spiking neural networks (event-driven RBP or eRBP) [30] and deep networks [39] on datasets such

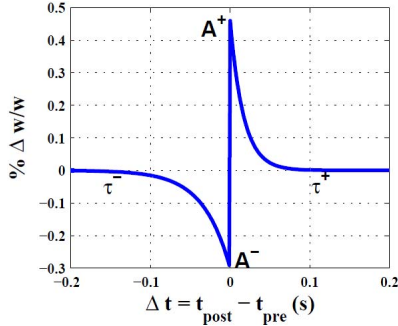


Fig. 2. Theoretical model of weight change as a function of time difference of pre- and post-synaptic spikes in STDP [21].

as MNIST, CIFAR-10, etc. Several variants of this algorithm have also been proposed such as skipped random backpropagation (SRBP), adaptive random backpropagation (ARBP), and adaptive skipped random backpropagation (ASRBP) [39].

C. Spike Timing Dependent Plasticity (STDP)

By far, the most popular learning algorithm to be implemented in recent adaptive neuromorphic systems is spike time dependent plasticity (STDP) [42], [43]. STDP has been widely observed in experiments [44], [45] as well as used in computational models [46], [47]. One popular mathematical model for STDP can be written as follows:

$$\Delta w = \begin{cases} \Delta w^+ = A^+ e^{-\frac{\Delta t}{\tau^+}}, & \text{if } \Delta t > 0 \\ \Delta w^- = A^- e^{\frac{\Delta t}{\tau^-}}, & \text{if } \Delta t \leq 0 \end{cases} \quad (7)$$

where $\Delta t = t_{post} - t_{pre}$. Essentially, this equation suggests that the synapse is potentiated (depotentiated) if post-synaptic spike happens after (before) the pre-synaptic one. A^+ and A^- govern the maximum weight change while τ^+ and τ^- dictate the time interval over which spiking activity is considered. Figure 2 plots the above equation. Since this rule only deals with pairs of spikes, it is sometimes referred to as pairwise or doublet STDP. Though the above formulation uses absolute change of weight, other formulations [48] use the same temporal dependence on relative weights ($\frac{\Delta w}{w}$).

Several variants of the simple STDP rule have been proposed to better match experimental data. One of these variants is the triplet rule [49] where the Δw depend not only on pre-post differences but pre-pre and post-post timing differences as well. It has been shown that augmenting the simple doublet rule with these added terms allows STDP to generalize to spike trains and behave similar to BCM learning rules [49]. FGMS and memristors, to be introduced in the following section III, have been used to implement STDP. Further, circuits to implement STDP in CMOS will be described in Section IV.

D. Spike Driven Synaptic Plasticity (SDSP)

The simple doublet STDP rule does not generalize well to spike train stimuli in learning experiments. Also, as mentioned earlier, it is unable to replicate biological measurements such as dominance of LTP at higher frequencies independent of spike timing differences. Hence, a more advanced rule has been proposed [35], [50] that takes into account post-synaptic depolarization ($V(t)$) as well as post-synaptic firing rates (captured in a Calcium variable $C(t)$). The synapses in this rule are bistable with weights of J_+ and J_- for potentiated and

depotentiated states respectively. However, there is an internal analog variable $X(t)$ that governs the transition between these states as shown in the following equations:

$$X = \begin{cases} X + a & \text{if } V(t_{pre}) > \theta_v \text{ and } \theta_{up}^l < C(t_{pre}) < \theta_{up}^h \\ X - b & \text{if } V(t_{pre}) \leq \theta_v \text{ and } \theta_{down}^l < C(t_{pre}) < \theta_{down}^h \end{cases} \quad (8)$$

where a and b are jump sizes, θ_v is voltage threshold (not related to neuron's spiking threshold) and θ_{up}^h , θ_{up}^l , θ_{down}^h and θ_{down}^l are thresholds on the calcium variable. The synaptic weight depends on this internal variable X in the following way:

$$w = \begin{cases} J_+ & \text{if } X > \theta_X \\ J_- & \text{if } X \leq \theta_X \end{cases} \quad (9)$$

In the absence of spikes, the variable X drifts to one of two states X_{max} or X_{min} depending on whether X is larger or smaller than θ_X . The corresponding equations are:

$$\frac{dX}{dt} = \begin{cases} \alpha & \text{if } X_{max} \geq X > \theta_X \\ -\beta & \text{if } X_{min} \leq X \leq \theta_X \end{cases} \quad (10)$$

One major advantage of this rule is that, since the synaptic weights are binary, it can be easily implemented in CMOS hardware [50]–[52] without using NVM devices. Example of such hardware is presented in Section IV-D. The dynamics of learning in such networks may be different from ones with high resolution weights especially in recurrent networks and needs to be carefully studied in future.

E. Structural Plasticity

While most learning algorithms in neuromorphic systems are related to the change in synaptic weights, the network structure of adult brains also undergoes considerable plasticity with new connections being formed and old ones being eliminated [53]–[55]. In contrast to weight plasticity, this mechanism is termed as structural plasticity (SP). The advantage of considering SP in neuromorphic systems is that presence or absence of connections can be indicated by 1-bit signals that are easy to store in CMOS latches with low write energy and cell area [52], [56]. Only recently these algorithms have been applied to real world classification problems [36], [38]. To overcome the loss in computational power due to 1-bit synapses, these works have used more complex neurons with multiple nonlinear dendrites. While [36], [38] applied the concept to non-spiking inputs and rate based spike trains, [57], [58] have applied SP to learn spike time based inputs. Both supervised [36], [38], [57], [59] and unsupervised versions [58], [60] of the SP algorithm have been presented. It has been also used in reservoir computing to improve the reservoir in an unsupervised fashion [60] and also to act as the readout layer [59].

The major concept of the SP algorithm may be summarized as follows:

- Create network with sparse synaptic connections—locations of each synapse are therefore important.
- Obtain Δw_{ij} using your supervised or unsupervised weight update algorithm of choice. For example, [36], [38] uses gradient descent, [58] uses STDP while [57] uses the tempotron [61] as the weight update algorithm. Also, hardware friendly approximations to the actual Δw_{ij} may be done [38], [59].

TABLE I
COMPARISON OF LEARNING ALGORITHMS USED IN RECENT NEUROMORPHIC MACHINE LEARNING SYSTEMS

Ref	Algorithm	Mode	Type	Accuracy	Precision (wt, internal)	Complexity	Locality	Convergence
[35]	SDSP	Spike	Semi-supervised	96.5% (MNIST) ^a	1-bit, high	Medium	Local	Problem Specific
[36]	Struct. plasticity	Spike	Supervised	96.4% (MNIST) ^b	1-bit, high	Medium	Global	Slow
[37]	STDP	Spike	Un-supervised	95% (MNIST) ^c	High	Low	Local	Problem Specific
[38]	Struct. plasticity	Non-Spike	Supervised	96/75/89% (UCI) ^d	1-bit, high	Medium	Global	Slow
[31]	Backprop (Pipelined)	Non-Spike	Supervised	98.07% (MNIST) ^e	8-bit, 2-bit	High	Global	Medium
[34]	Backprop (RUSD)	Non-Spike	Supervised	76%(CIFAR-10) 97.5% (MNIST) ^f	7-bit, 1-bit	Medium	Global	Slow
[39]	Backprop (RBP)	Non-Spike	Supervised	75%(CIFAR-10) 96% (MNIST) ^g	High, high	High	Global	Slow
[30]	Backprop (eRBP)	Spike	Supervised	98% (MNIST) ^h	High, high	high	Local	Slow
[19]	LMS (SOUL)	Non-spike	Supervised	95% (MNIST) ⁱ	15-bit, 15-bit	Low	Global	Fast.

^a SDSP is unsupervised but has been applied in this reference with a supervisory teacher signal. Internal variable is analog and precision requirement not specified. 115 neurons with $\approx 117K$ synapses used.

^b Neurons with dendritic nonlinearity used. Internal variable is analog and precision requirement not specified. 100 neurons with 3960 dendrites and $\approx 20K$ synapses used.

^c 6400 neurons with $\approx 500K$ synapses used. Adaptive threshold and conductance based models used.

^d Accuracies reported on BC, Heart and Ion datasets from UCI repository. [40]

^e 2 hidden layers with 600 neurons each

^f BinaryNet [41] with RUSD for CIFAR-10 and a CNN (2 conv. layers with 256 neurons each, 2 maxpooling layers and a fully connected layer) with RUSD for MNIST.

^g CNN (3 conv. layers with 64 neurons each, 3 max-pooling layers and a fully connected layer) with random backprop for CIFAR-10 and a DNN with 4 hidden layers (100 neurons each) for MNIST.

^h 2 hidden layers with 500 hidden nodes each

ⁱ 16,384 hidden neurons used, trained for 3 epochs of 60,000 training images.

- Instead of updating weight by adding Δw_{ij} , consider it as a fitness value C_{ij} . Synapses with low fitness values are candidates for replacement.
- Instead of random replacement, create a randomly selected candidate set for replacement. Evaluate fitness of these synapses in next iteration and choose the synapse with highest fitness value in the replacement set to replace the earlier selected synapse for replacement.
- Repeat above steps till some pre-defined convergence criteria is met.

One of the major advantages of this learning rule is the use of 1-bit synapses. Also, since the network has sparse synaptic connections, number of synapses used is far less compared to fully connected networks as can be seen in Table I. Also, changes of connections can be easily implemented in address event representation (AER) based spiking systems [51], [52], [62], [63] where a separate memory stores network connection details. While the inference part of this network has been implemented [64], an area of future focus should be implementation of online learning. However, the fitness value is still stored in high resolution (4-5 bits) in current algorithms [58] and needs to be reduced for more efficient VLSI implementation of online learning systems.

F. Others

1) *Online Update for Random Neural Networks*: Recently, a class of shallow neural networks with random weights in the

first layer have become popular due to their fast training speed, good generalization abilities and need for less training data. Termed extreme learning machine (ELM) [65] in the machine learning community, it has relations to earlier machine learning methods [66] as well as methods proposed in computational neuroscience [67], [68]. Compared to the reservoir computing methods [68], the major difference of ELM is the lack of feedback or recurrent connections. Due to the majority of weights in the network being random, it is very amenable to neuromorphic analog implementations [69]–[72]. Since only the second layer (output layer) of weights need to be trained, it requires very few weight updates. A supervised online training algorithm for this network is proposed in [73]; however, it requires many multiplications for each output weight update. Hence, a simplified version of this algorithm called Sign based Online Update Learning (SOUL) has been developed in [19]. The learning rule is given by the following equation:

$$\overline{\Delta w_{ij}} = \alpha \text{sgn}(e_i) \text{sgn}(h_j) \quad (11)$$

where w_{ij} is the weight connecting j-th hidden neuron to i-th output, $\text{sgn}()$ denotes the signum function, e_i is the error in the i-th output, h_j denotes output of the j-th hidden neuron and α is a learning rate. This update can be computed using a simple XOR gate and can be used to increment weights stored in counters. It can be shown [19] that this rule is related to a version of the LMS rule.

2) *Clustering-k-Means*: Clustering algorithms such as *k-means* are a class of popular unsupervised algorithms that are simple enough for low-power VLSI implementation [20]. In this method, the assumption is that the input data belongs to several classes with inter-class distance larger than intra-class distance. The objective of the learning algorithm is to learn the correct cluster centers (μ) and cluster spread (σ). Reference [20] present an FGMOS based implementation of this algorithm and achieves very high computational efficiency ($> 1\text{Tops/W}$).

3) *Sparse Coding-SAILnet*: With the seminal work of Olshausen and Field [74] that showed emergence of receptive fields similar to biology when sparse representation constraints are applied, a number of works have been devoted to exploring neural architectures for finding optimal sparse codes [75] and dictionaries [76]. The advantage of having sparse activations from a hardware viewpoint is that few neurons will spike in the network leading to less communication and hence less energy consumption. The algorithm proposed in [76] is particularly attractive for hardware implementation due to local weight update rules. It has been implemented in [77] to obtain low energy consumption in exchange for a small loss in classification accuracy.

G. Discussion

We present a comparison of the various algorithms on machine learning tasks in Table I. The column denoted as precision describes the precision of weights required by the algorithm. Also, some algorithms such as SDSP and structural plasticity use a low precision weight while requiring a higher precision internal variable—this is denoted by ‘internal’ in the same column. On the other hand, algorithms like RUSD and pipelined backpropagation use a higher precision weight while the internal variable is low precision.

Another important aspect is computational complexity of the weight update. Since there is no direct correspondence between non-spike methods such as backpropagation and spike based rules such as STDP, we instead compare the computational complexity per weight update (non-spike) or per spike (spike) in Table I. Though a direct comparison between a spike and a non-spike algorithm is difficult, it is feasible to compare the computational complexity of algorithms within spike or non-spike category.

Now, if a neural network has M layers and the number of neurons in each layer is n_i ($i = 1, 2..M$), for one backward pass the primary computations for weight update are multiplications. The number of such multiplications is given by:

$$C = \sum_{i=1}^{i=M-1} n_i \times n_{i+1} \quad (12)$$

Though there will be more operations required to add biases, compute gradients etc., it will not change scaling. So, the computational complexity for backpropagation will be $\mathcal{O}(C)$. Computational complexity of RBP and eRBP is same as regular backpropagation. For the pipelined backpropagation approaches, the computation time and hardware complexity is reduced by parallelizing the layer-wise computations and reducing the memory fetches. For RUSD, larger weight updates are applied with low probability. So, even though the theoretical complexity is same as backpropagation, actual number of computations is scaled down by a factor of p , the learning probability.

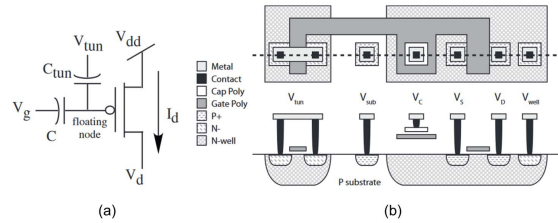


Fig. 3. Circuit schematic (a) and layout (b) (taken from [78]) of a floating-gate MOS (FGMOS). A FGMOS has all signals coupled to its gate through capacitors. The charge stored on the gate can be modified by quantum mechanical processes of electron tunneling and hot electron injection.

The SOUL algorithm uses weight updates only in the penultimate layer. So, if the number of output nodes is K and number of hidden nodes is L , the number of multiplications required is $K \times L$. Moreover, each multiplication here is equivalent to a bitwise XOR operation which further reduces the hardware cost of implementing it.

Among spike based algorithms, STDP employs a local learning rule. So, if a post-synaptic spike happens in a neuron and there are N pre-synaptic neurons connected to it, the N synaptic weight updates are to be computed. For SDSP, though the number of weights updated per post-synaptic spike is same as STDP, additional hardware complexity is required to calculate the internal analog variable $X(t)$ as described in [50].

The weight update rule is local for STDP, SDSP and eRBP while other algorithms use global weight update rules. For the SOUL algorithm, the convergence speed is fast as the weight update is applied to a single layer while for backpropagation variants the convergence is relatively slower. For the unsupervised and semi-supervised methods, it is difficult to compare the speed of convergence as it is highly dependent on the specific problem and the dataset characteristics.

III. NOVEL DEVICES

In this section, we review some of the novel devices that have been proposed over the years to perform the functions of neurons, synapses or adaptive elements, in general. Without losing generality, the relationship between the activity patterns of the input neurons \mathbf{x} and the output neurons \mathbf{y} in a NN can be expressed as:

$$\mathbf{y}_n = \mathbf{W}_{n \times m} \mathbf{x}_m \quad (13)$$

Most of the novel devices have been used to implement the synaptic function that is denoted by W in the equation 13. Some desirable properties of adaptive synapses are:

- **Non-volatile weight** storage is required so that the effect of learning is not lost with passage of time.
- **Compact size** is desired since the number of synapses is at least an order of magnitude larger than the number of neurons.
- **Low energy** operation for both creation of EPSC as well as for adaptation is required. In terms of non-volatile memory units used as synapses, these correspond to ‘read’ and ‘write’ energies.

A. Floating-Gate MOS

Historically, one of the earliest non-volatile storage elements proposed as a learning synapse was the floating-gate MOS (FGMOS) [79], [80]. As shown in Figure 3(a), a FGMOS has

TABLE II
COMPARISON OF FABRICATED FLOATING-GATE (FG) BASED ADAPTIVE ELEMENT DESIGN FOR NEUROMORPHIC SYSTEMS

Ref	Process (L in μm)	Mode	Algorithm	Energy	Area (μm^2) (divided by L^2)	Remarks
[96]	0.35	Spike	D-STDP	-	100 (816)	Single device measured
[97]	0.35	Spike	D-STDP	20 pJ/spike	133 (1088)	100 neurons, 10K fixed synapses, 20K plastic synapses
[21]	0.35	Spike	T-STDP	52 pJ/spike	6337 (51730)	Single device measured, can reduce to 0.52 pJ/spike by reducing C_g to 50 fF
[20]	0.13	Non-spike	k-means clustering	2 pJ/op	-	Op includes other analog operations apart from FG write
[101]	0.5	Non-spike	LMS	-	-	2 FGMOS based correlating synapse measured
[24]	0.35	Non-spike	LMS	1 pJ/op	-	48-tap, adaptive FIR filter with digital input and analog output with 10 bits of resolution

a regular PMOS but with no direct resistive connection to the gate. All signals are coupled capacitively to the gate of the transistor making it a floating node surrounded on all sides by high quality insulating oxide. The voltage on the floating node, V_{fg} depends on the stored charge on this node along with values of other coupling voltages. This stored charge can be modified by the quantum mechanical processes of hot electron injection (HEI) and Fowler-Nordheim tunneling [81]. Learning or adaptive systems using FGMOS modify this stored charge according to some pre-defined algorithm.

Due to its compatibility with CMOS, FGMOS devices have been integrated into various adaptive neuromorphic circuits in the past such as competitive learning [82]–[84], offset calibration in amplifiers, comparators and voltage references [85]–[89], adaptive filters [24], [25], [90], removing fixed pattern noise in imagers [91], [92], field programmable analog arrays [93]–[95], synapses with spike time dependent plasticity (STDP) [21], [96]–[98] and others [99]. Due to the ability of using a single transistor as a learning synapse in neuromorphic systems and ability to integrate tightly with CMOS circuits [18], FGMOS is a good potential candidate for building large scale adaptive neuromorphic systems. A comparison of different uses of the FG device is presented in Table II. A comparison between FG and non-FG synapses is presented in [15].

For the recently popular case of STDP learning, we can analyze the energy per write by first noting that weight, w can be expressed as:

$$w = e \frac{\Delta V_{fg}}{U_T} \quad (14)$$

where ΔV_{fg} is the slow time scale change in the FG voltage [96], [98] and U_T denotes thermal voltage. Thus, for a maximum change in weight of 10%, $\Delta V_{fg} \approx 2.5$ mV at room temperature. Now, we can write the energy/write due to tunneling (E_{tun}) and injection (E_{inj}) in terms of ΔV_{fg} as follows:

$$E_{tun} = C_{tot} \Delta V_{fg} (V_{tun} - V_{fg}) \quad (15)$$

$$E_{inj} = \frac{C_{tot} \Delta V_{fg}}{\alpha} (V_{ds, inj}) \quad (16)$$

where C_{tot} denotes total capacitance on the gate, V_{tun} denotes the applied tunneling voltage, α denotes the injection efficiency of generation of hot-electrons [100] and $V_{ds, inj}$ is the high drain-source voltage applied for injection. It should be noted that α is a function of $V_{ds, inj}$. While it may be argued that part of this energy is needed anyway for normal

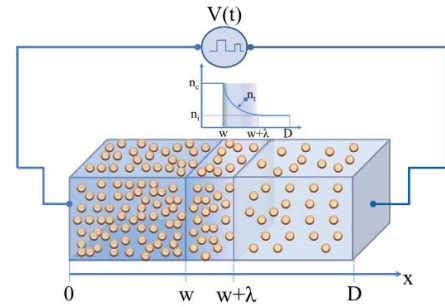


Fig. 4. Operating mechanism of a metal-oxide memristor. [109].

read operation (i.e. generation of EPSC), these equations still give us a method to estimate the energy/spike needed in FG synapses. As an example, by lowering C_{tot} to 50fF and keeping $V_{tun} - V_{fg} = 10V$, we can estimate the E_{tun} for 10% weight change to be only 1.25 fJ. E_{inj} is typically much larger (1.25–12.5pJ) due to small values of α ($\approx 10^{-3} - 10^{-4}$) [100].

B. Memristors

Emerging nonvolatile memory (NVM) denotes a series of new types of memory technologies that does not rely on electrical charge to store the data (e.g., as SRAM and DRAM). Some representative embedded NVM (eNVM) technologies are phase change memory (PCM) [102], spin-transfer-torque random-access-memory (STT-RAM) [103], resistive random-access-memory (RRAM) [104], ferroelectric field-effect-transistor (FeFET) memory [105] etc. Many of these eNVM technologies can be utilized to implement neuromorphic computing systems (NCS) where the programmable resistance of the eNVM cells represents the synaptic weights of the DNNs, especially RRAM (a.k.a. memristor). The first explicit theoretical depiction of memristor was given by Chua [106] in 1971 though earlier references to a similar element may be found in Widrow's work [107]. The resistance state (often referred to as memristance) of a memristor can be tuned by applying an electrical excitation. In 2008, HP Labs reported that memristive effect was realized by moving the doping front along a TiO_2 thin-film device [108], as conceptually illustrated in Figure 4. Here w , λ , and $(D-w-\lambda)$ are the lengths of conductive region, transition region, and insulating region, respectively. n_s is the electron density and the subscript $s = (c, t, i)$ denotes the parameter of these three regions.

Similarity between the programmable resistance state of memristors and the variable synaptic strengths of biological

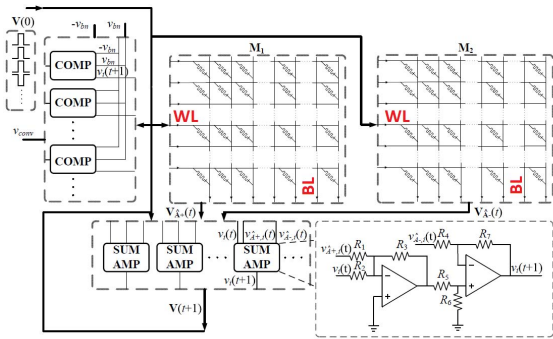


Fig. 5. Memristor-based NCS [110].

synapses dramatically simplifies the design of NCS. Figure 5 shows a memristor-crossbar-based NCS. With reference to equation 13, x is mimicked by the input voltage vector (or spikes) applied on the word-line (WL) of the memristor crossbar. Every memristor is programmed to the resistance state representing the weight of its corresponding synapse. The current on each bit-line (BL) of the memristor crossbar is collected and converted to the output voltage vector y by post-processing circuitry such as comparators or integrators. The $W_{n \times m}$ is often realized by two memristor crossbars, which respectively represents the positive and negative elements of the $W_{n \times m}$.

C. Spintronic Devices

Recent advancement in spintronic research has shown path towards ultra-low voltage, low current and energy-efficient computing beyond traditional CMOS. These devices exploit the new materials, device designs as well as novel switching phenomena such as spin-torque-transfer (STT), domain-wall (DW) movement, spin-hall effect (SHE), spin-orbital torque (SOT), and magnetoelectric (ME) switching [111]–[113]. These innovations not only augment energy-delay scaling roadmap in CMOS systems with disruptive solutions in energy-efficient power management, dense on-die non-volatile memories, non-volatile logic, etc., but also expand the computing paradigm with non-von Neumann architectures (e.g. neuromorphic hardware). In particular, explorations in spin-based neurons and spin-based synapses to mimic biological functionality in human brains have shown potential to achieve cognitive functionality comparable with CMOS digital hardware. They may also offer much less energy consumption close to biological operations owing to the “compact functionality” in one single device [111]. These neuron-synapse systems have been benchmarked with CMOS hardware in simulation, showing significant advantages in terms of power and energy-efficiency in the range of a few to hundreds of times improvement with desired recognition accuracy [114]–[116], [116]–[122]. In the following section, we discuss the recent progress in spin-based neurons and spin-based synapses, respectively. Note that spintronics has also been utilized to build nanoscale coupled oscillators for a different type of neuromorphic architecture [123], which will not be covered in the scope of this discussion.

1) *Spin Neurons*: Magnetic tunnel junctions (MTJ), well-known for on-die non-volatile memory applications, have also been investigated as the “spin neuron” candidate. A two-terminal MTJ is typically composed of two ferromagnetic layers separated by an oxide tunnel barrier. One layer has

fixed spin polarization, namely pinned layer (PL), and the other free layer (FL) can alter its polarization depending on the spin current through the MTJ via spin-transfer-torque (STT) effect. Depending on its parallel (P) or antiparallel state (AP) with respect to the PL, the resistance of the MTJ can be low resistance for P or high resistance for AP. Since the MTJ switching is governed by a critical current, the analogy to the thresholding effect in biological neuron (that accepts the weighted input from synapse) has been explored to build the biological neuron model.

Sharad *et al.* [114] proposed using a lateral spin valve (LSV), where an MTJ is connected to multiple domain wall magnet (DWM) based synapses (will be discussed later) and receives spin-weighted input current from a non-magnetic, metal channel. The metal channel sums the spin-polarized charge current and contacts the free layer of the MTJ. This current switches MTJ through nonlocal STT switching if it is above the critical value. The change of the MTJ resistance is then detected by a differential MTJ latch to generate a “spike” to the gate of a PMOS transistor that connects to the receiving synapse. Recently, Narasimman *et al.* [124] developed circuit solution to implement spike-timing-dependent-plasticity (STDP) based on this DWM-synapse and MTJ-neuron system by sampling pre- and post-synaptic time, showing on-line learning capability with significant energy-efficiency improvement.

Sharad *et al.* [115], [119] also proposed another approach using a three-terminal DWM with SHE assist and MTJ, where a free domain in DWM strip forms an MTJ with a tunnel barrier and a fixed magnet on top. Two fixed domains in opposite polarization states are separated by the free domain. During inference, the weighted and summed charge current from synapse crossbar connects to one terminal of the DWM (one of the fixed domain), which can move the DW depending on the current direction and thereby changes the MTJ resistance. This is sensed by a reference MTJ and connected to a CMOS inverter to generate a “spike”. The SHE layer underneath the DWM strip improves the DW velocity.

Besides LSV and DWM, Abhronil *et al.* [116] further proposed a SOT-based neuron using ferromagnet/heavy metal (FM/HM) device, where the free layer of the MTJ contacts the HM. The thresholding is achieved with 2-step control, where a clock current through HM to first orient MTJ FL to unstable point, then the write current through MTJ performs deterministic switching. The SOT effect by the clock current helps to minimize the required current and further improve the energy-efficiency.

With recent theoretical studies on probabilistic spiking nature of pyramidal neurons, Sengupta *et al.* [117], [118] employed the stochastic switching of MTJ as a function of current to mimic such neuron behavior. The write current (sum of the weighted input) through the HM to switch the MTJ follows a similar sigmoid probability density function, which (1) takes into account spintronic device variation nature and (2) allows mapping of graded analog transfer function in artificial neural network (ANN) to spiking neural network (SNN).

Given the binary switching nature of MTJ, the above approaches still fail to model the analog behavior of membrane potential. While leaky-integrate-fire (LIF) model has shown its capability to closely model neuron dynamics and been adopted in CMOS neuromorphic hardware, Jaiswal *et al.* proposed the LIF implementation using magnetoelectric

(ME)-oxide-ferromagnet heterostructure to model stochastic-LIF neuron [122]. Two transistors are added for synaptic input and leaky/reset, respectively, and the ME switching alters ferromagnet state. An MTJ formed by ferromagnet-MgO-pinned layer is used for reading and generating the spike with a reference MTJ and inverter. Given ME switching is charge-dependent and stochastic, the leaky-integrate-fire operation can be realized.

Another approach to achieve the analog neuron is to use multi-level MTJ [120]. Zhang *et al.* used a vertical stacked MTJ for both neuron and synapse to enable multi-step transfer function. Additional three NMOS transistors are used to perform reset, read and accepting synapse input, and one CMOS amplifier is used and MTJ is connected as feedback resistor (no need for reference MTJ). The higher the multi-level MTJ resistance, the higher the output voltage. Vatajelu and Anghel [121] further employed the similar configuration with MTJ but connecting the output spike voltage to synapse input for STDP modulation on synapse.

In all the described spin-neuron implementation, a “reset” is always required after firing in order to restore the nanomagnet to its reference state.

2) *Spin Synapses*: Besides the memristor-crossbar-based synapse, spintronic devices can also be used to implement synaptic operation. In [120], Zhang *et al.* used the vertical stacked multi-level MTJ for synapse which can achieve 4 different levels. Since increasing the number of level imposes changes on device fabrication, Vatajelu and Anghel [121] further proposed 2nd level crossbar connection for a single synapse (e.g. 3 × 3 connection with 9 MJT for a 10-level synapse), and showed the reduction in recognition error rate with improvement of synapse levels. DWM, owing to its displacement and spin-polarization current dependency, has also been used for synapse implementation. In all-spin based neuron-synapse system, Sharad *et al.* [114] used DWM to connect to spin neuron through metal channel, where the DW location is programmed off-line. The location of DW during inference modulates the spin-polarization current for spin neuron switching. However, DWM synapses may be limited by fanout number, spin diffusion length, number of levels (require long DWM strip), which are in research phase. Narasimman *et al.* [124] further implement STDP for DWM synapse to enable on-line learning. In [116], Abhronil *et al.* proposed a four-terminal ferromagnet-heavy metal heterostructure to implement STDP, which combines the MTJ with the DWM as free layer which contacts an HM. The learning is done through SOT by using the charge current through HM and applied magnetic field to modulate the DW displacement, while the reading is through MTJ and non-magnetic contact on the HM (decoupled current path for learning and reading). The transistors are connected to pre-spike and post-spike signals which modulate the DW based on the neuron firing activity. Comparing to memristor synapse approach, DWM synapse shows lower current operation due to the SHE enhanced spin injection efficiency which helps to reduce the power consumption per synaptic operation.

A summary of both spin neurons and synapses reported so far is presented in Table III. Comparing with CMOS-based approaches, spintronic devices offer opportunities to realize compact neuron and synapse models with a potential energy-efficiency improvement (e.g. lower supply voltage (<1V) and lower current compared to memristor and floating gate [124]–[126]). The low current requirements have a direct effect on noise and hence special stochastic algorithms might

be needed to train these networks [127]. Nevertheless, most of these devices still require CMOS circuits for signal propagation, control, sensing or functionality. Since most of the works are still in early research phase and simulation stage, main challenges remain in large-scale device demonstration including variation and reliability, novel material integration, spin injection efficiency improvement, etc and all improvements in simulation should be considered as potential gains only.

D. Discussion

Among the floating-gate circuits reported in Table II, the earliest ones were non-spiking ones where [24] used the same LMS algorithm as [101] but reduced technology node. Recent work [20] has used k-means as well as reduced technology node further to 0.13 μm CMOS. Among the STDP implementations, [96] was the first implementation of STDP on a FG device while [97] actually implemented an array with neurons and AER. Reference [21] implemented a more advanced version of the STDP rule including triplet spike interactions.

An important consideration for all adaptive elements to be used in neural network algorithms is intrinsic mismatch in programming desired weight values. Very few work have tried to explicitly correct the mismatch in a structured way, with [128] being one exception. In some specific cases, mismatch may be used in the computation [70] but these methods may require 2 – 3X more neurons. Moreover, they have not yet been shown for a general multi-layered network. FG based adaptive elements have been used for nearly three decades and have a much better control on statistical variations than their memristive counterparts. The usage of HEI as programming mechanism is also motivated by this fact since mismatch is much less than tunneling junctions [129]. Memristors typically show 2 – 3 bits of accuracy in large arrays [130] and only recently, there have been reports of ≈ 6 bits of accuracy [131]. In Table I, we show the precision requirement reported for some of the algorithms. An in-depth analysis of device mismatch in the context of these algorithms is an important direction for future work.

One attractive property for memristors over FGMOS devices is their potentially lower write energy. However, it is interesting to note that memristors suffer from poor selectivity issues in large arrays as a tradeoff. Typically, diodes are used to prevent sneak paths which lead to larger cell sizes. While the almost perfect selection of FGMOS using HEI makes system design simpler, we also see this as an opportunity to make better algorithms that are less affected by crosstalk and sneak paths in exchange for lower write energy.

Spin devices are a much more recent development with fewer published work and hence we have also reported neurons implemented in spintronic technology. For discussions on CMOS neurons, readers are referred to [132] and references therein.

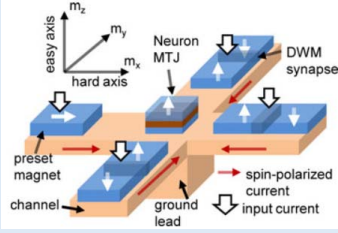
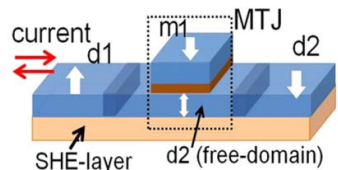
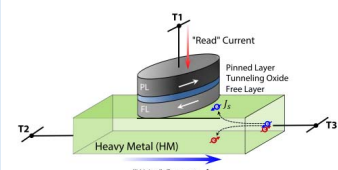
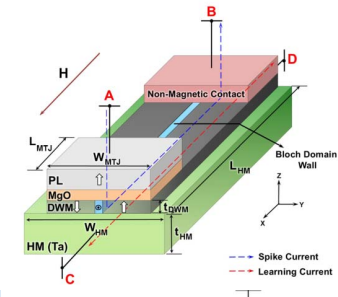
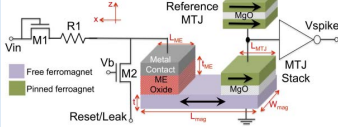
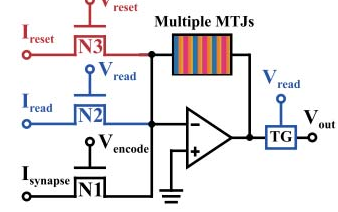
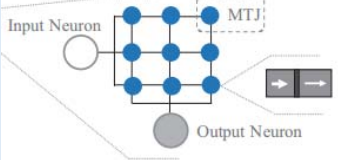
IV. CIRCUIT LEVEL INNOVATIONS

In this section, we review some of the unique circuit techniques devised to make compact, low-power adaptive circuits. Also, interfacing circuits required for enabling adaptation in large arrays of synaptic elements are discussed.

A. Transposable SRAM Synapse

A digital $N \times N$ SRAM crossbar structure can effectively represent a system of N neurons and N^2 possible

TABLE III
SUMMARY OF SPIN NEURON AND SYNAPSE DEVICES (SIMULATION)

Device Type	Circuit Topology	Energy	Accuracy	Applications	Cost
MTJ neuron and DWM synapse system [114], [124]		Average 51fJ/neuron [114]; 40fJ/weight update with online learning [124]	95.4% for handwriting digit recognition [124]	ANN implementation for pattern detection in MNIST	1 DWM per synapse, 1 MTJ per neuron, preset magnet and CMOS sensing/signaling circuits (8T with ref. MTJ). Additional STDP function write circuits (10T).
DWM neuron [115], [119]		48pJ for a single HTM level-2 node design [119]	Max 95% of matching threshold of 0.7 [119]	HTM inference hardware mapping with offline training	1 3-terminal DWM device per neuron, CMOS sensing circuits (1 INV or 7T latch with ref. MTJ). CMOS DAC and 5-bit SAR ADC.
MTJ neuron with SOT assist [117], [118]		19.5nJ per image operation [118]	83% for 0.2ns write time; 97% for 1ns write time [118]	Stochastic neuron; MNIST image recognition	1 3-terminal MTJ HM device per neuron, CMOS readout circuit (1 INV with ref. MTJ).
DWM synapse with SOT [116]		~2pJ per synaptic event [116]	N/A	STDP implementation	1 3-terminal MTJ DWM HM device with 4 CMOS access transistor per synapse and CMOS signaling post-neuron summing amplifier.
ME neuron [122]		246fJ per neuron per training iteration	80% with 1600 ME neurons	Stochastic LIF neuron implementation	1 ME with MTJ per neuron, CMOS signaling/sensing circuits (4T1R and ref MTJ).
Multi-level MTJ neuron and synapse [120]		N/A	91% with 60k training; 84% with 30% variation with 3 MTJ	Digital pattern recognition in MNIST	1 multi-level MTJ per neuron with CMOS signaling/sensing circuits (5T control circuits and 1 amplifier)
MTJ array synapse [121]		N/A	93% - 88% with 16 MTJ for min to max device-to-device variation	Digital pattern recognition in MNIST	Multiple MTJ per synapse (1 16), 1 MTJ per neuron with CMOS sensing/signaling circuits (7T control circuits, 1 amplifier, 1 pulse generator)

synaptic connections between them. Each row of the crossbar represents a neuron’s axon and each column represents a neuron’s dendrite. To efficiently implement time-dependent learning rules, both row and column accesses are important, since pre-synaptic row and post-synaptic column updates are required. Conventional memory arrays are accessed only in

rows, and column-based access would require inefficient serial row operations. Instead, this shortcoming is addressed in [56] by a transposable SRAM cell (Fig. 6) to store digital synapse weights, which enables single-cycle write and read access in both row and column directions to significantly enhance on-chip learning performance. The transposable SRAM cell

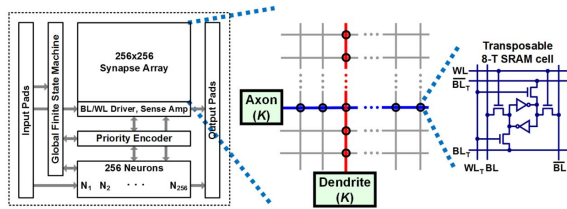


Fig. 6. A transposable SRAM cell design enables efficient row-by-row and column-by-column update for LTD and LTP update in synapse SRAM arrays [56].

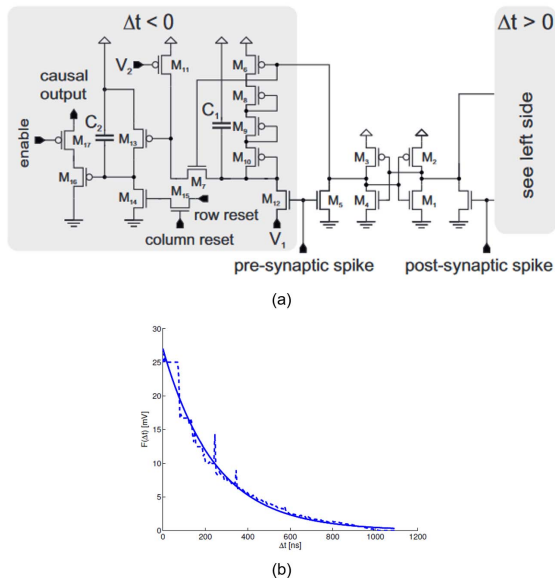


Fig. 7. (a) Circuit diagram of a MOSFET capacitor STDP synapse from [136] that uses a capacitor C_1 to compute present update and another capacitor C_2 to store accumulated updates. A central controller reads this value at regular intervals and updates the stored digital weight. (b) Measured weight update function is similar to the biological model in [48].

uses 8 transistors, adding two access transistors connected to word and bit lines in transposed orientations to a standard 6T design. For M -bit (2^M levels) synaptic weight storage, M transposable SRAM cells can be combined together to represent one synapse, operating with a common word line.

B. MOSFET Capacitor Synapse

One of the common methods to store analog synaptic weight is as charge on a capacitor [133]–[135]. However, this has the problem of leakage due to CMOS switches. One approach to circumvent this problem is proposed in [136], where the charge on the capacitor is sampled at regular intervals to determine whether to decrement or increment a digitally stored weight value. The digital weight is converted to an analog input stimulus using a Digital-to-Analog converter (DAC). Similar approaches had been used earlier for non-spiking neural networks [137]–[139]; [136] extends this to spiking systems.

Figure 7 shows the synapse circuit used in this case [136]. Only the causal part of the circuit is shown since the anti-causal one is exactly similar. A pre-synaptic (or post-synaptic) spike toggles the state of the latch comprising transistors M_1 – M_4 and puts the causal (or anti-causal) part in measurement state. Capacitor C_1 is used to calculate the current weight update as a function of most recent pre-post time

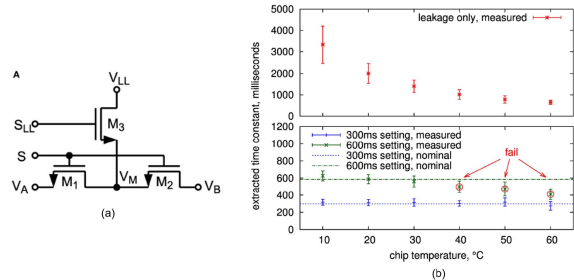


Fig. 8. (a) Circuit diagram of a low leakage switch used in a switched capacitor STDP synapse in 28 nm CMOS [52]. (b) Measured results show bio-realistic time constants of 600 ms achievable by this method.

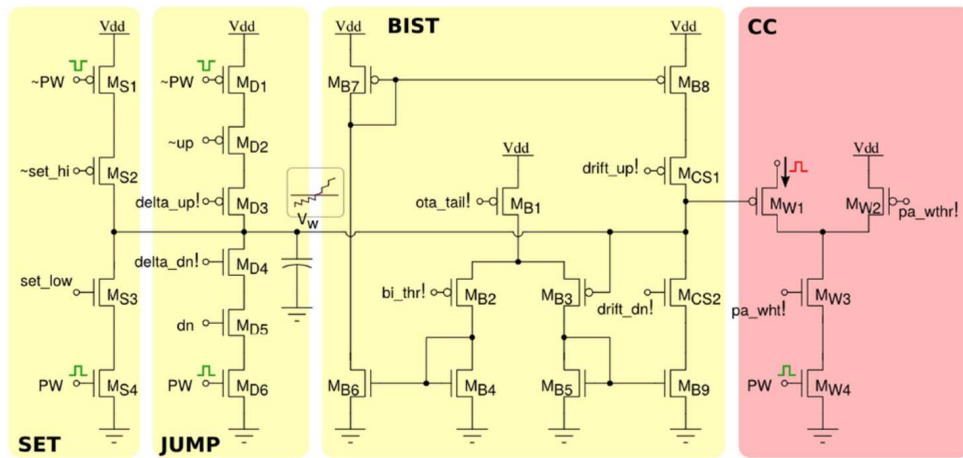
difference (Δt) while capacitor C_2 stores an accumulation of past updates. The voltage on C_2 is readout using the source follower transistor M_{16} and if it is larger than a threshold, the digital weight stored in SRAM is updated. Every pre-synaptic spike resets the voltage on C_1 to V_1 and from then on, the voltage leaks through the resistor string comprising sub-threshold transistors M_8 – M_{10} . A post-synaptic spike occurring after this cuts off this voltage leak by switching the latch state and thereby turning off M_6 . The remaining voltage on C_1 is indicative of $F(\Delta t)$ in the STDP equation. M_{13} is now used to add an amount of charge on C_2 that is proportional to the remaining voltage on C_1 . The synaptic resolution used in this work is 4 bits and it is shown to be sufficient for synchrony detection in an accompanying work [140]. Figure 7(b) shows the measured STDP curve matches the target function quite well.

C. Switched Capacitor Synapse

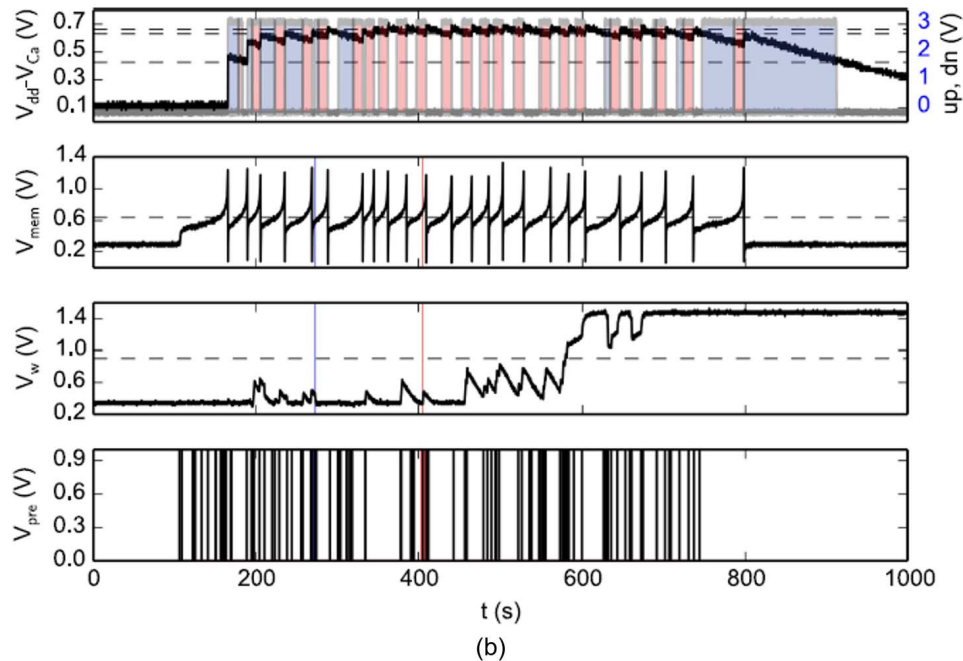
Switched capacitor circuits [141] have been used in the past to implement synaptic function [142], [143] as well as large-scale neuromorphic chips with different neuronal dynamics [132]. Recently, switched capacitors have also been used to implement a discrete time version of the SDSP algorithm [52] with stop learning feature. One advantage of this method is the use of above threshold circuits thus eliminating the mismatch related issues [144] commonly faced in sub-threshold designs. Similar to the architecture in [136], a central controller is used to cycle through and update weights based on local temporal correlation calculators. Since the chip was designed in an advanced technology (28 nm), leakage currents in traditional MOSFET switches were a major design issue to obtain long bio-realistic time constants spanning hundreds of milliseconds. Hence, the authors propose a new low leakage switch without operational amplifiers as shown in Figure 8(a). The switch is connected between nodes marked V_A and V_B while S is the control signal. S_{LL} is a complementary control signal that connects the middle node V_M to a voltage V_{LL} when the switch is not conducting. Choosing V_{LL} equal to the common mode voltage reduces the drain source voltage drop thus minimizing sub-threshold leakage. Also, the leakage becomes independent of voltage at the other terminal of the switch. Measured data from the test chip shown in Figure 8(b) proves that the leakage is sufficiently reduced to get a time constant as long as 600 ms.

D. Bi-Stable Synapse

Motivated by the difficulty in storing multiple levels of synaptic weights over long time scales, there has been work



(a)


 Fig. 9. (a) Circuit diagram of a SDSP synapse [51]. (b) Measured results show input spikes (bottom waveform), dynamics of the synaptic internal variable X in equation (9) (second waveform from bottom) and post-synaptic spikes (second waveform from top).

in developing algorithms and circuits for bi-stable plastic synapses [62], [63], [145], [146]. Example of such an algorithm is SDSP described in section II-D. The basic concept in these approaches is to have an internal analog state variable in the synapse (similar to X in equation (8)) that is stored as a charge on a capacitor. The value of this variable is modified according to ongoing spiking activity in the network. However, the weight of the synapse visible to the network is only one of two values (see equation (9)). This is obtained by operating a positive feedback circuit to latch to one of two states based on the value of this internal analog variable. Thus, even if the charge on the capacitor leaks out over long time periods, the synaptic weight remains fixed based on the last sustained activity profile of the network. This approach has been used to implement SDSP [146], SDSP with stop learning [51], [62] and STDP [63], [145].

Figure 9(a) shows an example of a circuit in [51] implementing the above dynamics. The circuits in the block marked SET are used to initialize the synapse to one of two states. The JUMP block implements equation (9) where the voltage on the capacitor, V_w , represents the variable X in the equation. Finally, the block BIST implements the positive feedback operation in equation (10). Among the waveforms in Figure 9(b), the second one from the bottom show dynamics of the variable X and how it has up and down jumps based on input spikes. Also, the final value is retained at a high state even after pre-synaptic spikes cease due to the latching effect of the positive feedback circuit marked BIST. The top waveform shows the calcium variable C in equation (9) based on post-synaptic activity integrated over a long time. It governs when to increase X (red shaded area), decrease X (blue shaded area) or to leave it unchanged.

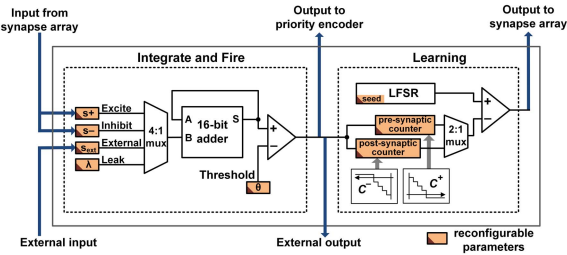


Fig. 10. Digital CMOS neuron with reconfigurability for STDP based learning implementation [56].

E. Interface Circuits

1) *Digital SRAM Interface Circuits*: Prior works have implemented digital CMOS circuits on the periphery of SRAM arrays that hold synapse weights. On-chip synaptic learning was demonstrated with binary synapses in [63], but the STDP circuitry was needed for each synapse element, which caused large circuit overhead. This was improved in [56], where synapse learning circuitry was shared across axons (rows) and dendrites (columns) at the SRAM crossbar periphery. The neuron block diagram that integrates the STDP learning circuitry is shown in Fig. 10. 8-bit time-keeping counters $C+$ and $C-$ enable independent control of pre-synaptic and post-synaptic updates, respectively, by tracking the time elapsed since the last spiking event of each neuron.

2) *RRAM Interface Circuits*: Interface circuits for memristor crossbar arrays were presented in a parallel architecture with resistive crosspoint array (PARCA) [147], [148], as summarized in Figure 11. The crosspoint array (D) connects a vector (z) on the row side interface and another vector (r) on the column side interface. Key operations that can be fully parallelized are: Dz and D update. To compute Dz in PARCA, the read process of the memristor array is utilized. For every z , a small read voltage V_z is applied simultaneously for every non-zero binary bit of z . V_z is multiplied with the conductance G at each crosspoint, and the weighted sum results in the output current at each r node. Compared to conventional memories that require row-by-row read operation, such multi-row access reads the entire crossbar array in parallel, thereby accelerating Dz . Furthermore, weight updates for the memristor crossbar array are performed through a write operation, by local programming voltage generation at local z and r nodes. Such approach can implement the spike-based learning algorithms or variants of stochastic gradient descent, where the weight value change is mapped to the conductance value change of memristor devices [147], [148].

3) *Spin-Based Interface Circuits*: Spin-based interface circuits have also been proposed to assist the operation of spin neurons and synapses. For example, to evaluate the MTJ based neuron state (firing or not), a reference MTJ in antiparallel state was connected in series with the MTJ neuron and input to an inverter. When the MTJ neuron is in antiparallel state, the inverter output is low; while MTJ neuron is switched to parallel state, the inverter output is high [115], [116], [116]–[122]. For all-spin based neuron-synapse system, a differential latch with MTJ was used to latch the output and drive the next spin synapse [115]. Depending on the spintronic device operation, CMOS circuits are usually required to provide clocking and control of the signal propagation in the neural nets.

V. NEUROMORPHIC ARCHITECTURES

In this section, we present details of commonly used architectures in large scale neuromorphic systems and discuss their relative advantages and disadvantages.

A. Crossbar Synaptic Array

The most common architecture in neuromorphic systems is the crossbar where an array of $N \times M$ synaptic elements connect N input neurons with M output neurons. As shown in Figure 12, the N inputs are presented along the rows while the M outputs are obtained along the columns. Thus, each crosspoint must have a synaptic element, which, in this case is shown to be a memristor. Do note that crossbars were used widely [79] even before memristors and examples of CMOS crossbar [97] will be discussed later.

Extensive studies in leveraging memristor to build new form of computing mechanism in crossbars have been conducted. For example, the theoretical concept and verification of using memristor crossbar structure to perform matrix-vector computation was firstly proposed in 2012 [102], [110]. The spiking based circuit implementation for neural networks in Figure 12 [149] and a system-level demonstration [102] with TiO_2 memristor devices were realized. At the architectural level, the memristor crossbars are extended to build general reconfigurable neuromorphic computing accelerators [150] or the components for some specific applications, such as approximated computation [151] and convolutional neural network [152], [153]. In these designs, both weight storage and matrix-based operations are completed within the crossbars. A design automation framework was also developed for large scale neuromorphic systems. Through pruning, permutation, and partitioning, a large network can be mapped to many crossbars for computation in an efficient way [154]. The optimization methodologies on power, area and accuracy of the relevant systems were also widely explored [155].

In implementing on-chip neuromorphic computing system, training methods have been proposed to program the memristor devices in a crossbar by following existing training algorithms in neural network models. For example, Li *et al.* introduced a mixed-signal self-training acceleration framework [156]. As illustrated in Figure 13, for a training task of an N -layer neural network with $(N - 1)$ weight matrices, the training acceleration framework requires $(N - 1)$ normal crossbars and $(N - 2)$ copy crossbar arrays. An array of subtractors is used to work out the deviation (δ_p) between the actual and ideal output and the error of node k in the next neighbor layer (δ_k) will be calculated through the copy crossbar arrays.

To understand the impact of device imperfections and circuit design constraints on the training robustness, Liu *et al.* [157] conducted a quantitative analysis of two popular training methods and proposed Vortex – a variation-aware training scheme. Vortex can actively compensate the impact of device variations and optimize the mapping scheme from computations to crossbars, therefore, leading to enhanced training robustness. Figure 14 is a 44×44 pattern in a 64×64 1T1R array programmed and measured by HP Labs [158]. It can be seen that the single-bit failure (SBF) defects distribute randomly across the array and blur the programmed pattern. According to the low yield of current memristor array development, Liu *et al.* proposed a defect rescuing methodology which improves the hardware efficiency by leveraging the

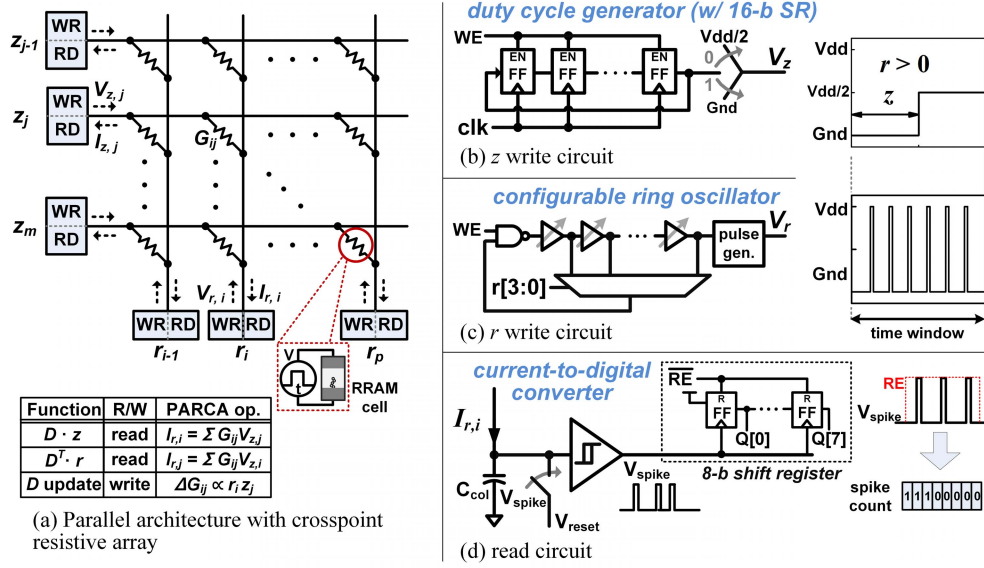


Fig. 11. (a) Parallel architecture of resistive crosspoint array (PARCA) [147]. (b) Write circuits for duty cycle window generation. (c) Write circuits that generate a number of pulses that are equally spaced over time. (d) Read circuits that convert the output current (weighted sum) into digital values.

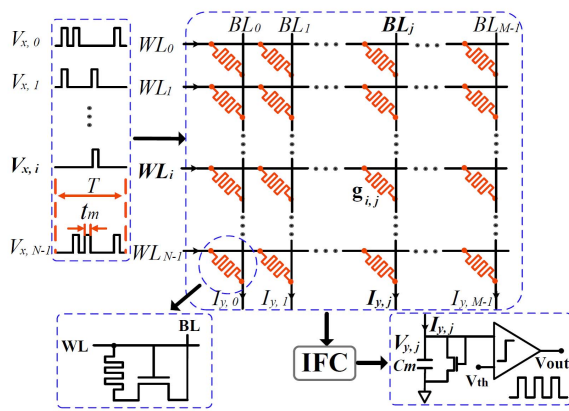


Fig. 12. An overview of the spiking neuromorphic design with a resistive crossbar array [149].

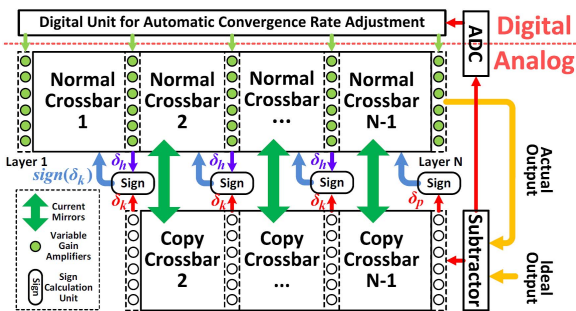


Fig. 13. Mixed-signal training acceleration framework [156].

application-specific features [158]: it first classifies the weights in a neural network into significant and insignificant categories based on their impacts on the network performance; then a retraining algorithm was developed to compensate the SBF caused computation error by re-tuning the trainable weights.

Theoretically, a memristor can be programmed to any arbitrary resistance state. In reality, however, the programming

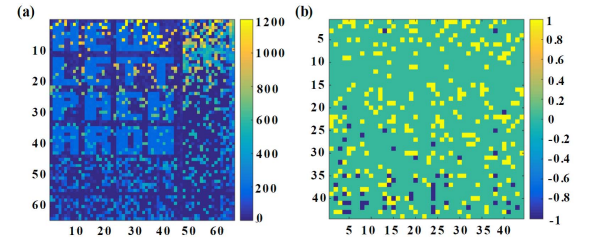


Fig. 14. (a) The conductance distribution and (b) the measured stuck on/off defects of a 64×64 memristor cross-point array [158].

process is limited by the resolution that CMOS circuitry can offer. In memristor-based crossbar based neuromorphic architecture, for instance, limited programming resolution requires a *quantization* process that maps each analog weight to one of the values that are represented by the discrete resistance states of the memristors: an analog weight within the range between a_i and a_{i+1} ($i = 0, \dots, m-1$) in the neural network are represented by only one value $a_{Li} \in [a_i, a_{i+1}]$ after quantization. Here m is the number of distinctive levels that the resistance of memristors can be programmed to. To minimize the impact of quantization loss, Wang *et al.* [159] proposed to discretize weights in different layers different values and then directly learn a neural network with discrete weights. More generally, Song *et al.* [160] introduce cosine and sawtooth regularization to binary and three-level representations to the memristor-based neuromorphic systems.

Scalability is another challenge in neuromorphic system development. It is inevitable to interconnect multiple crossbars to implement modern big neural networks. The increasing scale of neural networks could quickly exhaust the resources of synapse crossbars and deteriorate the wire congestion. The framework Group Scissor [161] attempts to save hardware area and improve system scalability through two steps: *rank clipping* integrates low-rank approximation into the training to reduce total crossbar area and *group connection deletion* structurally prunes connections to reduce routing congestion between crossbars.

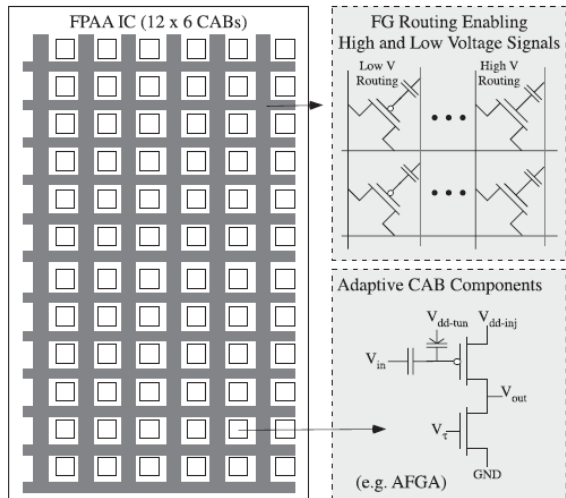


Fig. 15. Island style routing architecture employed in FPAA where neurons or computing elements are in “islands” within a sea of routing switches.

Crossbar architectures are also very popular in CMOS implementations. For example, Brink *et al.* [97] implemented a cross-bar of 100×300 synapses of which 100×100 were fixed weight, 100×100 were recurrent excitatory learning synapses and the remaining 100×100 were learning synapses connected to input. Another example is the 256×256 crossbar array of SDSP synapses reported in [51].

B. Island Style

The major issue with a crossbar style hardware architecture is that the number of synaptic connections increases as $O(N^2)$ for N neural elements. While this is the most flexible architecture, it has a huge requirement of hardware resources. Typical biological networks have a different architecture where local interconnections are dense but long distance connections are sparse [162]. In keeping with that philosophy, some neuromorphic systems [93], [94], [163] have used an island style architecture that is commonly observed in field programmable gate arrays (FPGA). As shown in Figure 15, the computational elements are organized in “islands” called computational analog blocks (CAB) while there is a big “sea” of switches for routing. Termed as field programmable analog array (FPAA) in keeping with their digital counterparts, one major difference of FG MOS based FPAA, however, is that the FG MOS switches can be used for computation and need not be only ON/OFF switches. As an example, they have been used as synaptic weights [164].

While most earlier FPAAs needed to be programmed separately for new applications or weight updates [93], [164], a newly introduced one [163] has runtime adaptation capability. Competitive neural circuits and LMS adaptation rules have been demonstrated on this platform by continuous time FG adaptation [101], [165]. Special care was taken in this design to allow routing of high voltage signals required for tunneling and hot-electron injection.

C. Crossbar With Time-Multiplexed Update (TMU)

This is a special architecture [52], [140] devised to take advantage of the low-power, low-area parallel processing capabilities of analog systems and the multi-bit long-term

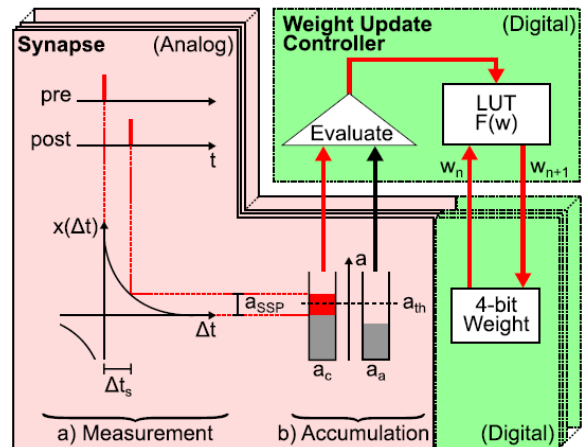


Fig. 16. A mixed signal scheme for analog multi-bit synapses where an analog crossbar computes correlations in parallel and stores charge on a capacitor. A controller sequentially reads these values and updates multi-bit weights in digital memory (from [140]).

storage available easily in digital memory. Here, the temporal correlations required for STDP are calculated in parallel in a synaptic crossbar and the result is stored in a capacitor. As shown in Figure 16, a digital controller sequentially reads these values and updates the corresponding weight in a digital SRAM according to the desired temporal update rule $F(\Delta t)$. While applying inputs to the system, these digital weights are again read and applied to the analog neurons through digital-analog converters (DAC). While this method does enable multi-bit weight, it is not scalable to large crossbar due to the large readout delays in such systems which will cause large leakage induced errors in the capacitor voltage. An island style architecture would be needed to scale this approach to bigger systems.

D. Discussion

A comparison of different recently reported neuromorphic systems are presented in Table IV. Amongst the ones implementing SDSP, [62] is the earliest while [51] has ported the design to a more advanced node of $0.18\mu\text{m}$ CMOS and increased the number of synapses by an order of magnitude. [52] also use SDSP but the focus of their design is to achieve biological time scales in leakage dominated process nodes like 28nm. 1-bit STDP implementations were first done on a large scale in [63]. Next, 4-bit implementations were shown in [136] using a time multiplexed approach. Finally, even higher resolution of ≈ 10 bits were demonstrated in [97] using FG MOS devices. While these used mixed signal approaches, an alternative fully digital design methodology was adopted in [56] for ease of porting across technology nodes. A similar digital design approach is used in [77] where a different algorithms based on sparse activations is used to reduce communication energy further.

VI. APPLICATIONS

In this section, we discuss some of the emerging applications that will benefit from adaptive neuromorphic hardware.

A. Brain Machine Interface (BMI)

Brain-machine interfaces is an emerging area of research where an electronic implantable or wearable device is used

TABLE IV
COMPARISON OF RECENTLY FABRICATED ADAPTIVE NEUROMORPHIC SYSTEMS

Ref	Process (μm)	Mode	Algorithm	Adaptive Element	Architecture	Remarks
[97]	0.35	Spike	D-STDP	FGMOS	Crossbar	100 neurons, 10K fixed synapses, 20K plastic synapses with ~ 10 -bit resolution.
[20]	0.13	Non-spike	k-means clustering	FGMOS	Full-custom	Implements Deep Machine learning feature extraction with 3 layer, 7 node network.
[163]	0.35	Non-spike	LMS, Competition	FGMOS	Island	14×6 array of CABs with FGMOS, capacitor, transistor, OTA in each CAB. Further, around 100K FGMOS in switch matrix.
[136]	0.18	Spike	STDP	Capacitor, SRAM	Crossbar (TMU)	384 neurons and $\sim 98K$ synapses with 4-bit synaptic weights stored in SRAM.
[52]	0.028	Spike	SDSP	Capacitor, SRAM	Crossbar (TMU)	64 neurons and $\sim 8K$ synapses with 1-bit resolution, switched capacitor design.
[51]	0.18	Spike	SDSP	Capacitor, Latch	Crossbar	256 neurons and 128K synapses with 1-bit resolution.
[63]	0.25	Spike	STDP	Capacitor, Latch	Crossbar	1024 neurons, $\sim 21K$ synapses with 1-bit resolution, current mode design.
[166]	0.35	Spike	STDP	Capacitor	Crossbar	32 neurons with 2K synapses, volatile weight stored as charge on capacitor.
[62]	0.35	Spike	SDSP	Capacitor, latch	Crossbar	16 neurons with $\sim 2K$ plastic synapses with 1-bit resolution, 48 fixed weight excitatory and inhibitory synapses each.
[56]	0.045	Spike	STDP	SRAM	Crossbar	256 neurons, 64K synapses with 1-bit resolution, full digital design.
[77]	0.065	Spike	Sparse coding (SAILnet)	SRAM	Ring and crossbar	256 neurons, 128K synapses with 8-13 bit resolution, fully digital design.

to provide a direct means of communication to the outside world for a patient who has lost other means of communication due to an accident or illness such as stroke [167]. While wearable devices do not have a risk of infection, they provide very coarse information about the brain state. On the other hand, implantable devices provide more fine grained information down to single unit activity which is needed for decoding dexterous motor skills [168]. Most of the published work so far has attempted to integrate the neural recording amplifier [169], ADC and wireless transmitter in the implant while keeping the motor intention decoding [170] on an external PC. At most, some work integrate a spike detector [171] or a spike sorter [172] in the implanted chip.

However, with the increase in number of channels on the microelectrode array (MEA), we are approaching a big-data transmission problem in neural implants [173]. Adaptive neuromorphic spike based decoders are a natural and elegant solution to this problem. It can directly mimic the part of the brain that would decode spikes to produce a command for movement in a healthy individual and has potential to become a natural interface. Adaptation is necessary here since the neural signals are non-stationary and change with time due to shift in position of the implant or due to formation of scar tissue [174], [175]. There are only a few published work on neural decoder circuits [175]–[177], but we believe there is great potential for neuromorphic systems to impact this application and break the inherent tradeoff between number of channels in MEA versus wireless transmission capacity. Reference [175], [177] present a neuromorphic implementation of a randomized neural network that can be trained with very few samples to tackle the retraining problem due to non-stationarity. Reference [176] presents an alternative solution where the neuromorphic system adapts learns in real-time from the biological spike train.

B. Robotics

We are supposed to be at the cusp of the fourth industrial revolution with robotics expected to play a big role. While it is expected that artificial intelligence powered by deep learning may enable future robots to be able to be much more efficient than its predecessors, this is one area where clearly taking inspiration from biology will help since humans can do many tasks very intuitively which are very difficult for robots. It is possible that this is related to the fundamentally different ways in which signals are represented in human brains. To mimic biology more closely, neuromorphic visual [178], [179], auditory [180] and tactile [181] sensors have been developed which operate in an asynchronous, spike based manner similar to human retina, cochlea and mechanoreceptors, respectively. This asynchronous sampling results in most informative part of the signal being readout first leading to quick reaction times [182]–[184]. Also, some problems that are difficult to solve with traditional sampling methods often have elegant solutions in this asynchronous spike domain [185]. We expect that work on adaptive spike based neuromorphic hardware to interface with these sensors is an important future direction for robotics, especially those with tight constraints on size, weight and power, such as drones. One example is shown in [186] where a neuromorphic processor [51] is used for obstacle avoidance and target acquisition for a land robot.

C. Internet of Things (IoT)

The internet of things aims to connect billions of devices with the cloud—sensors collect data and transmit to the cloud for processing. But this method is likely not scalable due to the huge burden that wireless transmission puts on the battery life [187], [188], especially for data intensive sensors like camera or microphone. Moreover, the data deluge on the cloud leads to an information overload and might be

impossible for the cloud to process and give results in time critical applications. This has led to the concept of “Edge Computing” [189]—moving information processing and refinement to the sensor node instead of the cloud. It has a huge market potential and is projected to grow to \$6.7B by 2022 [190]. For edge computing to succeed, there is a need for adaptive machine learning systems that can be low-power to sustain long battery lives while refining raw data to low-bandwidth information. Reference [20] presents an adaptive neuromorphic system that reduces dimensionality of data by extracting features. These features can now be transmitted at lower energy cost or maybe further used by a classifier on the sensor node to make a final decision. As another example, consider the problem of predictive maintenance [191] where the adaptive system needs to track the health condition of the machine and raise an early warning in case of a problem. This requires a different model to be learnt for every machine and necessarily requires an adaptive neuromorphic approach. We foresee edge computing devices to be a big market for neuromorphic engineering.

VII. CONCLUSION

In this paper, we presented a survey of work done in the area of adaptive neuromorphic systems with a focus on on-chip learning. We saw that while FGMOS devices have the advantage of compatibility with CMOS, RRAM devices have the potential to scale to even smaller sizes. Spin mode devices such as domain wall memory have the potential for most energy efficient designs but no measured results have been reported from spin mode neuromorphic systems.

In terms of learning algorithms, various simplified versions of backpropagation are being developed for efficient hardware implementation. In particular, the feedback alignment method which requires only random weights for feeding back gradients seems particularly attractive due to its local nature. Also, shallow two layer networks based on random hidden nodes (ELM, RVFL, LSM, etc.) are attractive for adaptive systems due to their quick training. We also expect future work to focus on novel plasticity algorithms beyond weight based methods such as structural plasticity.

Popular adaptive synapse circuits in CMOS implement the SDSP rule since it only needs bistable weights in longer term. The most popular architecture for large scale systems is the crossbar topology due to its high packing density. However, for scaling to larger systems, island style architectures with dense local and sparse global connections are preferred. Finally, we expect these adaptive systems to find uses in several emerging applications such as brain machine interfaces and size, weight, power constrained robotics platforms such as micro aerial vehicles.

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, May 2015.
- [2] F.-F. Li, A. Karpathy, and J. Johnson. *Stanford CS Class CS231n: Convolutional Neural Networks for Visual Recognition*. Accessed: Oct. 15, 2017. [Online]. Available: <http://cs231n.stanford.edu/>
- [3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1106–1114.
- [4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR*, 2016, pp. 770–778.
- [5] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification,” in *Proc. CVPR*, 2014, pp. 1701–1708.

- [6] G. Hinton *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
- [7] L. Deng *et al.*, “Recent advances in deep learning for speech research at Microsoft,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2013, pp. 8604–8608.
- [8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Nov. 2011.
- [9] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [10] C. Mead, “Neuromorphic electronic systems,” *Proc. IEEE*, vol. 10, no. 10, pp. 1629–1636, Oct. 1990.
- [11] *Top 10 Emerging Technologies of 2015*. Accessed: Oct. 19, 2017. [Online]. Available: <https://www.weforum.org/agenda/2015/03/top-10-emerging-technologies-of-2015-2/>
- [12] *Global Neuromorphic Chip Market*. Accessed: Oct. 19, 2017. [Online]. Available: <https://www.transparencymarketresearch.com/pressrelease/neuromorphic-chip-market.html>
- [13] *Neuromorphic Chip Market Worth US 1.8 Billion by 2023*. Accessed: Oct. 20, 2017. [Online]. Available: <https://www.marketwatch.com/story/neuromorphic-chip-market-worth-us18-billion-by-2023-tmr-2017-05-18-82033139>
- [14] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *CoRR*, vol. abs/1703.09039, pp. 2295–2329, Dec. 2017. [Online]. Available: <http://arxiv.org/abs/1703.09039>
- [15] M. R. Azghadi, N. Iannella, S. F. Al-Sarawi, G. Indiveri, and D. Abbott, “Spike-based synaptic plasticity in silicon: Design, implementation, application, and challenges,” *Proc. IEEE*, vol. 102, no. 5, pp. 717–737, May 2014.
- [16] G. Indiveri and S.-C. Liu, “Memory and information processing in neuromorphic systems,” *Proc. IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015.
- [17] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, “Neuromorphic electronic circuits for building autonomous cognitive systems,” *Proc. IEEE*, vol. 102, no. 9, pp. 1367–1388, Sep. 2014.
- [18] J. Hasler and B. Marr, “Finding a roadmap to achieve large neuromorphic hardware systems,” *Frontiers Neurosci.*, vol. 7, p. 118, Sep. 2013.
- [19] C. S. Thakur *et al.* “An online learning algorithm for neuromorphic hardware implementation.” [Online]. Available: <https://arxiv.org/abs/1505.02495>
- [20] J. Lu, S. Young, I. Arel, and J. Holleman, “A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 μm CMOS,” *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 270–281, Jan. 2015.
- [21] R. Gopalakrishnan and A. Basu, “Triple spike time-dependent plasticity in a floating-gate synapse,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 4, pp. 778–790, Apr. 2017.
- [22] B. Widrow and M. A. Lehr, “30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation,” *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, Sep. 1990.
- [23] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.
- [24] M. Figueroa, S. Bridges, D. Hsu, and C. Diorio, “A 19.2 GOPS mixed-signal filter with floating-gate adaptation,” *IEEE J. Solid-State Circuits*, vol. 39, no. 7, pp. 1196–1201, Jul. 2004.
- [25] Y. Zhai and P. A. Abshire, “Adaptive log domain filters for system identification using floating gate transistors,” *Analog Integr. Circuits Signal Process.*, vol. 56, nos. 1–2, pp. 23–36, 2008.
- [26] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, DC, USA: Spartan, 1962.
- [27] C. H. Mays, “Adaptive threshold logic,” Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 1963.
- [28] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, “Random synaptic feedback weights support error backpropagation for deep learning,” *Nature Commun.*, vol. 7, Nov. 2016, Art. no. 13276.
- [29] S. Grossberg, “Competitive learning: From interactive activation to adaptive resonance,” *Cognit. Sci.*, vol. 11, no. 1, pp. 23–63, 1987.
- [30] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, “Event-driven random back-propagation: Enabling neuromorphic deep learning machines,” *Frontiers Neurosci.*, vol. 11, p. 324, Jun. 2017.
- [31] H. Mostafa, B. Pedroni, S. Sheik, and G. Cauwenberghs, “Hardware-efficient on-line learning through pipelined truncated-error backpropagation in binary-state networks,” *Frontiers Neurosci.*, vol. 11, p. 496, Sep. 2017.
- [32] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.

- [33] L. K. Muller and G. Indiveri. (Apr. 2015). "Rounding methods for neural networks with low resolution synaptic weights." [Online]. Available: <https://arxiv.org/abs/1504.05767>
- [34] L. K. Müller, M. V. Nair, and G. Indiveri, "Randomized unregulated step descent for limited precision synaptic elements," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
- [35] J. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Comput.*, vol. 19, no. 11, pp. 2881–2912, 2007.
- [36] S. Hussain and A. Basu, "Multiclass classification by adaptive network of dendritic neurons with binary synapses using structural plasticity," *Frontiers Neurosci.*, vol. 10, p. 113, Mar. 2016.
- [37] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers Neurosci.*, vol. 9, p. 99, Aug. 2015.
- [38] S. Hussain, S.-C. Liu, and A. Basu, "Hardware-amenable structural learning for spike-based pattern classification using a simple model of active dendrites," *Neural Comput.*, vol. 27, no. 4, pp. 845–897, 2015.
- [39] P. Baldi, P. Sadowski, and Z. Lu. (Dec. 2016). "Learning in the machine: Random backpropagation and the deep learning channel." [Online]. Available: <https://arxiv.org/abs/1612.02734>
- [40] M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [41] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. (Feb. 2016) "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1." [Online]. Available: <https://arxiv.org/abs/1602.02830>
- [42] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: Taming the beast," *Nature Neurosci.*, vol. 3, no. 11, pp. 1178–1183, 2000.
- [43] H. Markram, W. Gerstner, and P. J. Sjöström, "Spike-timing-dependent plasticity: A comprehensive overview," *Front. Synaptic Neuroscience*, vol. 4, no. 2, 2012.
- [44] R. Froemke and Y. Dan, "Spike-timing-dependent synaptic modification induced by natural spike trains," *Nature*, vol. 416, no. 6879, pp. 433–438, 2002.
- [45] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, 1998.
- [46] N. Iannella and S. Tanaka, "Synaptic efficacy cluster formation across the dendrite via STDP," *Neurosci. Lett.*, vol. 403, nos. 1–2, pp. 24–29, 2006.
- [47] N. Iannella, T. Launey, and S. Tanaka, "Spike timing-dependent plasticity as the origin of the formation of clustered synaptic efficacy engrams," *Front. Comput. Neurosci.*, vol. 4, p. 21, Jul. 2010.
- [48] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neurosci.*, vol. 3, no. 9, pp. 919–926, 2000.
- [49] J.-P. Pfister and W. Gerstner, "Triplets of spikes in a model of spike timing-dependent plasticity," *J. Neurosci.*, vol. 26, no. 38, pp. 9673–9682, Sep. 2006.
- [50] S. Fusi, M. Annunziato, D. Badoni, A. Salamon, and D. J. Amit, "Spike-driven synaptic plasticity: Theory, simulation, VLSI implementation," *Neural Comput.*, vol. 12, no. 10, pp. 2227–2258, 2000.
- [51] N. Qiao *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers Neurosci.*, vol. 9, p. 141, Apr. 2015.
- [52] M. Noack *et al.*, "Switched-capacitor realization of presynaptic short-term-plasticity and stop-learning synapses in 28 nm CMOS," *Frontiers Neurosci.*, vol. 9, p. 10, Feb. 2015.
- [53] G. W. Knott, A. Holtmaat, L. Wilbrecht, E. Welker, and K. Svoboda, "Spine growth precedes synapse formation in the adult neocortex *in vivo*," *Nature Neurosci.*, vol. 9, no. 9, pp. 1117–1124, 2006.
- [54] D. B. Chklovskii, B. W. Mel, and K. Svoboda, "Cortical rewiring and information storage," *Nature*, vol. 431, no. 7010, pp. 782–788, 2004.
- [55] M. Butz, F. Wörgötter, and A. V. Ooyen, "Activity-dependent structural plasticity," *Brain Res. Rev.*, vol. 60, no. 2, pp. 287–305, 2009.
- [56] J.-S. Seo *et al.*, "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Sep. 2011, pp. 1–4.
- [57] S. Roy, P. P. San, S. Hussain, L. W. Wei, and A. Basu, "Learning spike time codes through morphological learning with binary synapses," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 7, pp. 1572–1577, Jul. 2016.
- [58] S. Roy and A. Basu, "An online unsupervised structural plasticity algorithm for spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 4, pp. 900–910, Apr. 2017.
- [59] S. Roy, A. Banerjee, and A. Basu, "Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 5, pp. 681–695, Oct. 2014.
- [60] S. Roy and A. Basu, "An online structural plasticity rule for generating better reservoirs," *Neural Comput.*, vol. 28, no. 11, pp. 2557–2584, 2016.
- [61] R. Gutig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nature Neurosci.*, vol. 6, no. 3, pp. 420–428, 2006.
- [62] S. Mitra, S. Fusi, and G. Indiveri, "Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI," *IEEE Trans. Biomed. Circuits Syst.*, vol. 3, no. 1, pp. 32–42, Feb. 2009.
- [63] J. V. Arthur and K. Boahen, "Learning in silicon: Timing is everything," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2006, pp. 75–82.
- [64] A. Banerjee, S. Kar, S. Roy, A. Bhaduri, and A. Basu, "A current-mode spiking neural classifier with lumped dendritic nonlinearity," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 714–717.
- [65] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.
- [66] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.
- [67] C. Eliasmith and C. H. Andersen, *Neural Engineering: Computation, Representation and Dynamics in Neurobiological Systems*. Cambridge, MA, USA: MIT Press, 2003.
- [68] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.
- [69] A. Basu, S. Shuo, H. Zhou, M. H. Lim, and G.-B. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," *Neurocomputing*, vol. 102, pp. 125–134, Feb. 2012.
- [70] E. Yao and A. Basu, "VLSI extreme learning machine: A design space exploration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 60–74, Jan. 2017.
- [71] C. S. Thakur, R. Wang, T. J. Hamilton, J. Tapson, and A. V. Schaik, "A low power trainable neuromorphic integrated circuit that is tolerant to device mismatch," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 2, pp. 211–221, Feb. 2016.
- [72] A. Patil, S. Shen, E. Yao, and A. Basu, "Hardware architecture for large parallel array of random feature extractors applied to image recognition," *Neurocomputing*, vol. 261, pp. 193–203, Oct. 2017.
- [73] J. Tapson and A. van Schaik, "Learning the pseudoinverse solution to network weights," *Neural Netw.*, vol. 45, pp. 94–100, Sep. 2013.
- [74] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, Jun. 1996.
- [75] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Comput.*, vol. 20, no. 10, pp. 2526–2563, 2008.
- [76] J. Zylberberg, J. T. Murphy, and M. R. DeWeese, "A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields," *PLoS Comput. Biol.*, vol. 7, no. 10, p. e1002250, 2011.
- [77] P. Knag, J. K. Kim, T. Chen, and Z. Zhang, "A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding," *IEEE J. Solid-State Circuits*, vol. 50, no. 4, pp. 1070–1079, Apr. 2015.
- [78] C. M. Twigg, "Floating gate based large-scale field-programmable analog arrays for analog signal processing," Ph.D. dissertation, School Electr. Comput. Eng., Georgia Inst. Technol., Atlanta, GA, USA, 2006.
- [79] P. Hasler, C. Diorio, B. Minch, and C. Mead, "Single transistor learning synapses," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 1994, pp. 817–824.
- [80] C. Diorio, P. Hasler, B. A. Minch, and C. Mead, "A floating-gate MOS learning array with locally computed weight updates," *IEEE Trans. Electron Devices*, vol. 44, no. 12, pp. 2281–2289, Dec. 1997.
- [81] A. Basu and P. E. Hasler, "A fully integrated architecture for fast and accurate programming of floating gates over six decades of current," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 6, pp. 953–962, Jun. 2011.
- [82] D. Hsu, M. Figueroa, and C. Diorio, "Competitive learning with floating-gate circuits," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 732–744, May 2002.

- [83] P. Hasler, "Continuous-time feedback in floating-gate MOS circuits," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 56–64, Jan. 2001.
- [84] P. Hasler, P. Smith, C. Duffy, C. Gordon, J. Dugger, and D. Anderson, "A floating-gate vector-quantizer," in *Proc. IEEE 45th Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2002, pp. 196–199.
- [85] P. Hasler, B. A. Minch, and C. Diorio, "An autozeroing floating-gate amplifier," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 74–82, Jan. 2001.
- [86] Y. L. Wong, M. H. Cohen, and P. A. Abshire, "A 1.2-GHz comparator with adaptable offset in 0.35- μ m CMOS," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 9, pp. 2584–2594, Oct. 2008.
- [87] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision CMOS amplifier using floating-gate transistors for offset cancellation," *IEEE J. Solid-State Circuits*, vol. 42, no. 2, pp. 280–291, Feb. 2007.
- [88] F. Adil, G. Serrano, and P. Hasler, "Offset removal using floating-gate circuits for mixed-signal systems," in *Proc. Outhwest Symp. Mixed-Signal Design*, 2003, pp. 190–195.
- [89] V. Srinivasan, G. Serrano, C. M. Twigg, and P. Hasler, "A floating-gate-based programmable CMOS reference," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3448–3456, Dec. 2008.
- [90] M. Kucic, A. Low, P. Hasler, and J. Neff, "A programmable continuous-time floating-gate Fourier processor," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 90–99, Jan. 2001.
- [91] Y. L. Wong and P. A. Abshire, "A 144 \times 144 current-mode image sensor with self-adapting mismatch reduction," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 8, pp. 1687–1697, Aug. 2007.
- [92] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "MATIA: A programmable 80 μ w/frame CMOS block matrix transform imager architecture," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, Mar. 2006.
- [93] A. Basu *et al.*, "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sep. 2010.
- [94] C. R. Schlottmann, S. Shaper, S. Nease, and P. Hasler, "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE J. Solid-State Circuits*, vol. 47, no. 9, pp. 2174–2184, Sep. 2012.
- [95] S. George *et al.*, "A programmable and configurable mixed-mode FPAA SoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2253–2261, Jun. 2016.
- [96] S. Ramakrishnan, P. E. Hasler, and C. Gordon, "Floating gate synapses with spike-time-dependent plasticity," *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 3, pp. 244–252, Jun. 2011.
- [97] S. Brink *et al.*, "A learning-enabled neuron array IC based upon transistor channel models of biological phenomena," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 1, pp. 71–81, Feb. 2013.
- [98] R. Gopalakrishnan and A. Basu, "On the non-STDP behavior and its remedy in a floating-gate synapse," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2596–2601, Oct. 2015.
- [99] C. Diorio, D. Hsu, and M. Figueroa, "Adaptive CMOS: From biological inspiration to systems-on-a-chip," *Proc. IEEE*, vol. 90, no. 3, pp. 345–357, Mar. 2002.
- [100] P. Hasler, A. Basu, and S. Koziol, "Above threshold pFET injection modeling intended for programming floating-gate systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 1557–1560.
- [101] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 834–845, May 2005.
- [102] H. Li and Y. Chen, *Nonvolatile Memory Design: Magnetic, Resistive, and Phase Change*. Boca Raton, FL, USA: CRC Press, 2011.
- [103] Y. Chen, X. Wang, H. Li, H. Xi, Y. Yan, and W. Zhu, "Design margin exploration of spin-transfer torque RAM (STT-RAM) in scaled technologies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 12, pp. 1724–1734, Dec. 2010.
- [104] Y. Chen, W. Tian, H. Li, X. Wang, and W. Zhu, "PCMO device with high switching stability," *IEEE Electron Device Lett.*, vol. 31, no. 8, pp. 866–868, Aug. 2010.
- [105] I. Bayram, E. Eken, X. Wang, X. Sun, T. P. Ma, and Y. Chen, "Adaptive refreshing and read voltage control scheme for FeDRAM," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 1154–1157.
- [106] L. O. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.
- [107] B. Widrow, W. H. Pierce, and J. B. Angell, "Birth, life, and death in microelectronic systems," Office Naval Res., Stanford Univ., Stanford, CA, USA, Tech. Rep. 1552-2/1851-1, May 1961.
- [108] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, May 2008.
- [109] L. Zhang, Z. Chen, J. J. Yang, B. Wysocki, N. McDonald, and Y. Chen, "A compact modeling of TiO₂-TiO_{2-x} memristor," *Appl. Phys. Lett.*, vol. 102, no. 15, p. 153503, 2013.
- [110] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1864–1878, Oct. 2014.
- [111] N. Locatelli *et al.*, "Spin torque nanodevices for bio-inspired computing," in *Proc. 14th Int. Workshop Cellular Nanosc. Netw. Their Appl. (CNNA)*, Jul. 2014, pp. 1–2.
- [112] G. W. Burr *et al.*, "Neuromorphic computing using non-volatile memory," *Adv. Phys.*, X, vol. 2, no. 1, pp. 89–124, 2017.
- [113] Y. Chen, X. Wang, H. Li, H. Liu, and D. V. Dimitrov, "Design margin exploration of spin-torque transfer RAM (SPRAM)," in *Proc. 9th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2008, pp. 684–690.
- [114] M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy, "Spin-based neuron model with domain-wall magnets as synapse," *IEEE Trans. Nanotechnol.*, vol. 11, no. 4, pp. 843–853, Jul. 2012.
- [115] M. Sharad, D. Fan, and K. Roy, "Spin-neurons: A possible path to energy-efficient neuromorphic computers," *J. Appl. Phys.*, vol. 114, no. 23, p. 234906, Dec. 2013.
- [116] A. Sengupta, S. H. Choday, Y. Kim, and K. Roy, "Spin orbit torque based electronic neuron," *Appl. Phys. Lett.*, vol. 106, no. 14, p. 143701, 2015.
- [117] A. Sengupta, P. Panda, P. Wijesinghe, Y. Kim, and K. Roy, "Magnetic tunnel junction mimics stochastic cortical spiking neurons," *Scientific reports*, vol. 6, p. 30039, 2016.
- [118] A. Sengupta, M. Parsa, B. Han, and K. Roy, "Probabilistic deep spiking neural systems enabled by magnetic tunnel junction," *IEEE Trans. Electron Devices*, vol. 63, no. 7, pp. 2963–2970, Jul. 2016.
- [119] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1907–1919, Sep. 2016.
- [120] D. Zhang *et al.*, "All spin artificial neural networks based on compound spintronic synapse and neuron," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 4, pp. 828–836, Apr. 2016.
- [121] E. I. Vatajelu and L. Anghel, "Fully-connected single-layer STT-MTJ-based spiking neural network under process variability," in *Proc. IEEE/ACM Int. Symp. Nanosc. Archit. (NANOARCH)*, Jul. 2017, pp. 21–26.
- [122] A. Jaiswal, S. Roy, G. Srinivasan, and K. Roy, "Proposal for a leaky-integrate-fire spiking neuron based on magnetoelectric switching of ferromagnets," *IEEE Trans. Electron Devices*, vol. 64, no. 4, pp. 1818–1824, Apr. 2017.
- [123] J. Torrejon *et al.*, "Neuromorphic computing with nanoscale spintronic oscillators," *Nature*, vol. 547, pp. 428–431, Jul. 2017.
- [124] G. Narasimman, S. Roy, X. Fong, K. Roy, C.-H. Chang, and A. Basu, "A low-voltage, low power STDP synapse implementation using domain-wall magnets for spiking neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 914–917.
- [125] A. Sengupta and K. Roy, "A vision for all-spin neural networks: A device to system perspective," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 12, pp. 2267–2277, Dec. 2016.
- [126] A. Sengupta, Z. A. Azim, X. Fong, and K. Roy, "Spin-orbit torque induced spike-timing dependent plasticity," *Appl. Phys. Lett.*, vol. 106, no. 9, p. 093704, 2015.
- [127] A. Sengupta and K. Roy, "Encoding neural and synaptic functionalities in electron spin: A pathway to efficient neuromorphic computing," *Appl. Phys. Rev.*, vol. 4, no. 4, p. 041105, 2017.
- [128] S. Shuo and A. Basu, "Analysis and reduction of mismatch in silicon neurons," in *Proc. IEEE Biomed. Circuits Syst. (BioCAS)*, Nov. 2011, pp. 257–260.
- [129] S. Kim, J. Hasler, and S. George, "Integrated floating-gate programming environment for system-level ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2244–2252, Jun. 2016.
- [130] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, "Sparse coding with memristor networks," *Nature Nanotechnol.*, vol. 12, pp. 784–789, May 2017.
- [131] M. Hu *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater.*, vol. 30, no. 9, p. 1705914, 2018.
- [132] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers Neurosci.*, vol. 5, p. 73, May 2011.
- [133] H. Tanaka, T. Morie, and K. Aihara, "A CMOS spiking neural network circuit with symmetric/asymmetric STDP function," *IEICE Trans. Fund. Electron. Commun. Comput. Sci.*, vol. E92A, no. 7, pp. 1690–1698, 2009.

- [134] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and STDP integrated circuits," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 3, pp. 246–256, Jun. 2012.
- [135] T. J. Koickal, A. Hamilton, S. L. Tan, J. A. Covington, J. W. Gardner, and T. C. Pearce, "Analog VLSI circuit implementation of an adaptive neuromorphic olfaction chip," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 1, pp. 60–73, Jan. 2007.
- [136] J. Schemmel, A. Grubl, K. Meier, and E. Mueller, "Implementing synaptic plasticity in a VLSI spiking neural network model," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2006, pp. 1–6.
- [137] G. Cauwenberghs, "Analog VLSI long-term dynamic storage," in *Proc. IEEE Symp. Circuits Syst. (ISCAS)*, May 1996, pp. 334–337.
- [138] P. Hasler and L. A. Akers, "A continuous time synapse employing a refreshable multilevel memory," in *Proc. Seattle Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 1, Jul. 1991, pp. 563–568.
- [139] S. Satyanarayana, Y. P. Tsvividis, and H. Graf, "A reconfigurable analog VLSI neural network chip," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 1990, pp. 758–768.
- [140] T. Pfeil *et al.*, "Is a 4-bit synaptic weight resolution enough?—Constraints on enabling spike-timing dependent plasticity in neuromorphic hardware," *Frontiers Neurosci.*, vol. 6, p. 90, Jul. 2012.
- [141] P. E. Allen and D. R. Holberg, *CMOS Analog Circuit Design*, 2nd ed. London, U.K.: Oxford Univ. Press, 2002.
- [142] Y. Tsvividis, "Switched neural networks," U.S. Patent 4873 661, Oct. 10, 1989.
- [143] M. Walker, P. Hasler, and L. A. Akers, "A CMOS neural network for pattern association," *IEEE Micro*, vol. 9, no. 5, pp. 68–74, Oct. 1989.
- [144] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1212–1224, Jun. 2005.
- [145] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 211–221, Jan. 2006.
- [146] E. Chicca *et al.*, "A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1297–1307, Sep. 2003.
- [147] J.-S. Seo *et al.*, "On-chip sparse learning acceleration with CMOS and resistive synaptic devices," *IEEE Trans. Nanotechnol.*, vol. 14, no. 6, pp. 969–979, Nov. 2015.
- [148] D. Kadetotad *et al.*, "Parallel architecture with resistive crosspoint array for dictionary learning acceleration," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 194–204, Jun. 2015.
- [149] C. Liu *et al.*, "A spiking neuromorphic design with resistive crossbar," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.
- [150] X. Liu *et al.*, "RENO: A high-efficient reconfigurable neuromorphic computing accelerator design," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.
- [151] B. Li, Y. Shan, M. Hu, Y. Wang, Y. Chen, and H. Yang, "Memristor-based approximated computation," in *Proc. Int. Symp. Low Power Electron. Design*, Sep. 2013, pp. 242–247.
- [152] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. 43rd Int. Symp. Comput. Archit.*, Jun. 2016, pp. 14–26.
- [153] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 541–552.
- [154] W. Wen *et al.*, "An EDA framework for large scale hybrid neuromorphic computing systems," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.
- [155] B. Li, L. Xia, P. Gu, Y. Wang, and H. Yang, "MERging the interface: Power, area and accuracy co-optimization for RRAM crossbar-based mixed-signal computing system," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.
- [156] B. Li, Y. Wang, Y. Wang, Y. Chen, and H. Yang, "Training itself: Mixed-signal training acceleration for memristor-based neural network," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jun. 2014, pp. 361–366.
- [157] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-aware training for memristor X-bar," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.
- [158] C. Liu, M. Hu, J. P. Strachan, and H. H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *Proc. 54th Annu. Design Autom. Conf.*, 2017, Art. no. 87.
- [159] Y. Wang, W. Wen, L. Song, and H. H. Li, "Classification accuracy improvement for neuromorphic computing systems with one-level precision synapses," in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 776–781.
- [160] C. Song, B. Liu, W. Wen, H. Li, and Y. Chen, "A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system," in *Proc. IEEE 6th Non-Volatile Memory Syst. Appl. Symp. (NVMSA)*, c Aug. 2017, pp. 1–6.
- [161] Y. Wang, W. Wen, B. Liu, D. Chiarulli, and H. H. Li, "Group scissor: Scaling neuromorphic computing design to large neural networks," in *Proc. 54th Annu. Design Autom. Conf.*, 2017, Art. no. 85.
- [162] P. A. Merolla *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.
- [163] S. Brink, J. Hasler, and R. Wunderlich, "Adaptive floating-gate circuit enabled large-scale FPAA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 11, pp. 2307–2315, Nov. 2014.
- [164] A. Basu, S. Ramakrishnan, C. Petre, S. Koziol, S. Brink, and P. E. Hasler, "Neural dynamics in reconfigurable silicon," *IEEE Trans. Biomed. Circuits Syst.*, vol. 4, no. 5, pp. 311–319, Oct. 2010.
- [165] P. Hasler and J. Dugger, "Correlation learning rule in floating-gate pFET synapses," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 65–73, Jan. 2001.
- [166] S. A. Bamford, A. F. Murray, and D. J. Willshaw, "Spike-timing-dependent plasticity with weight dependence evoked from physical constraints," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 4, pp. 385–398, Aug. 2012.
- [167] M. Lebedev *et al.*, "Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface," *J. Neurosci.*, vol. 25, no. 19, pp. 493–4681, 2005.
- [168] S. Acharya, F. Tenore, V. Aggarwal, R. Etienne-Cummings, M. H. Schieber, and N. V. Thakor, "Decoding individuated finger movements using volume-constrained neuronal ensembles in the M1 hand area," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 16, no. 1, pp. 15–23, Feb. 2008.
- [169] Y. Chen *et al.*, "A digitally assisted, signal folding neural recording amplifier," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 8, pp. 528–542, Aug. 2014.
- [170] J. C. Kao, S. D. Stavisky, D. Sussillo, P. Nuyujukian, and K. V. Shenoy, "Information systems opportunities in brain-machine interface decoders," *Proc. IEEE*, vol. 102, no. 5, pp. 666–682, May 2014.
- [171] E. Yao, Y. Chen, and A. Basu, "A 0.7 V, 40 nW compact, current-mode neural spike detector in 65 nm CMOS," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 2, pp. 309–318, Apr. 2016.
- [172] V. Karkare, S. Gibson, C.-H. Yang, H. Chen, and D. Marković, "A 75 μ W, 16-channel neural spike-sorting processor with unsupervised clustering," in *IEEE Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2011, pp. 252–253.
- [173] A. Basu, Y. Chen, and E. Yao, "Big data management in neural implants: The neuromorphic approach," in *Emerging Technology and Architecture for Big-Data Analytics*, A. Chattopadhyay, C. H. Chang, and H. Yu, Eds. Cham, Switzerland: Springer, 2017.
- [174] D. Sussillo, S. D. Stavisky, J. C. Kao, S. I. Ryu, and K. V. Shenoy, "Making brain-machine interfaces robust to future neural variability," *Nature Commun.*, vol. 7, Dec. 2016, Art. no. 13749.
- [175] S. Shaikh, Y. Chen, R. So, and A. Basu, "Cortical motor intention decoding on an analog co-processor with fast training for non-stationary data," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Jul. 2017, pp. 294–297.
- [176] F. Boi *et al.*, "A bidirectional brain-machine interface featuring a neuromorphic hardware decoder," *Frontiers Neurosci.*, vol. 10, Dec. 2016, Art. no. 563.
- [177] Y. Chen, E. Yao, and A. Basu, "A 128-channel extreme learning machine-based neural decoder for brain machine interfaces," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 3, pp. 679–692, Jun. 2016.
- [178] M. Yang, S.-C. Liu, and T. Delbruck, "A dynamic vision sensor with 1% temporal contrast sensitivity and in-pixel asynchronous delta modulator for event encoding," *IEEE J. Solid-State Circuits*, vol. 50, no. 9, pp. 2149–2160, Sep. 2015.
- [179] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 10, pp. 259–275, Jan. 2010.
- [180] M. Yang, C.-H. Chien, T. Delbruck, and S.-C. Liu, "A 0.5V 55 μ W 64 \times 2 channel binaural silicon cochlea for event-driven stereo-audio sensing," *IEEE J. Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, Nov. 2016.
- [181] M. Rasouli, Y. Chen, A. Basu, N. V. Thakor, and S. L. Kukreja, "Spike-based tactile pattern recognition using an extreme learning machine," in *Proc. IEEE Biomed. Circuits Syst. (BioCAS)*, Oct. 2015, pp. 1–4.

- [182] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas, and T. Delbruck, "A pencil balancing robot using a pair of AER dynamic vision sensors," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2009, pp. 781–784.
- [183] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Frontiers Neuroscience*, vol. 7, p. 223, Nov. 2013.
- [184] W. W. Lee, S. L. Kukreja, and N. V. Thakor, "Discrimination of dynamic tactile contact by temporally precise event sensing in spiking neuromorphic networks," *Frontiers Neurosci.*, vol. 11, p. 5, Jan. 2017.
- [185] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous event-based binocular stereo matching," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 347–353, Feb. 2012.
- [186] M. B. Milde *et al.*, "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system," *Frontiers Neurobot.*, vol. 11, p. 28, Jul. 2017.
- [187] F. Conti *et al.*, "An IoT endpoint system-on-chip for secure and energy-efficient near-sensor analytics," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2481–2494, Sep. 2017.
- [188] Z. Wang and N. Verma, "A low-energy machine-learning classifier based on clocked comparators for direct inference on analog sensors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 11, pp. 2954–2965, Nov. 2017.
- [189] *What is Edge Computing?* Accessed: Nov. 4, 2017. [Online]. Available: <https://www.ge.com/digital/blog/what-edge-computing>
- [190] *Market for Edge Computing?* Accessed: Nov. 2, 2017. [Online]. Available: <https://www.prnewswire.com/news-releases/edge-computing-market-worth-672-billion-usd-by-2022-654465673.html>
- [191] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine learning for predictive maintenance: A multiple classifier approach," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 812–820, Jun. 2015.



Arindam Basu (M'10) received the B.Tech. and M.Tech. degrees in electronics and electrical communication engineering from the IIT Kharagpur in 2005, and the M.S. degree in mathematics and the Ph.D. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, in 2009 and 2010, respectively. He joined Nanyang Technological University in 2010, where he currently holds a tenured associate professor position. He received the Prime Minister of the India Gold Medal in 2005 from IIT Kharagpur.

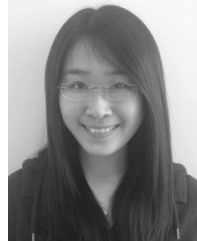
His research interests include bio-inspired neuromorphic circuits, non-linear dynamics in neural systems, low-power analog IC design, and programmable circuits and devices. He was a Distinguished Lecturer of the IEEE Circuits and Systems Society for the 2016–2017 term. He received the Best Student Paper Award from the Ultrasonics symposium in 2006, best live demonstration at ISCAS 2010 and a finalist position in the best student paper contest at ISCAS 2008. He was received the MIT Technology Review's inaugural TR35@Singapore Award in 2012 for being among the top 12 innovators under the age of 35 in Southeast Asia, Australia, and New Zealand. He was a Guest Editor for two special issues in the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS for selected papers from ISCAS 2015 and BioCAS 2015. He is serving as a Corresponding Guest Editor for the special issue on low-power, adaptive neuromorphic systems: devices, circuit, architectures and algorithms in the IEEE JOURNAL ON EMERGING TOPICS IN CIRCUITS AND SYSTEMS. He is currently an Associate Editor of the IEEE SENSORS JOURNAL, the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, and *Frontiers in Neuroscience*.



Jyotibha Acharya received the B.E. degree in ECE from Jadavpur University, India, in 2014. He is currently pursuing the Ph.D. degree with the Interdisciplinary Graduate School, Nanyang Technological University, Singapore. After completing his B.E. degree, he was with Philips Healthcare as a Graduate Engineer Trainee (imaging systems). His research interests include neuromorphic algorithms and low-power wearable biomedical devices.



Tanay Karnik (M'88–SM'04–F'13) received the Ph.D. degree in computer engineering from the University of Illinois at Urbana-Champaign. He was the Director of Intel's University Research Office. He joined Intel Corporation in 1995. He is currently a Principal Engineer with the Microarchitecture Research Lab, Intel Corporation. He has published over 80 technical papers, has 74 issued and 40 pending patents in these areas. His research interests include small form factor systems, 3-D architectures, variation tolerance, power delivery, and architectures for novel devices. He was a member of ISSCC, DAC, ICCAD, ICICDT, ISVLSI, ISCAS, 3DIC, and ISQED Program Committees, and JSSC, TCAD, TVLSI, TCAS Review Committees. He is an ISQED Fellow, an Associate Editor of TVLSI, and a Senior Advisory Board Member of JETCAS. He received the Intel Achievement Award for the pioneering work on integrated power delivery. He has presented several keynotes, invited talks, and tutorials, and has served on seven Ph.D. students' committees. He was a General Chair of ISLPED'14, ASQED'10, ISQED'09, ISQED'08, and ICICDT'08. He was a Guest Editor of JSSC.



Huichu Liu (S'11–M'15) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2009, and the Ph.D. degree in the electrical engineering from Pennsylvania State University, University Park, PA, USA, in 2014.

She interned at the T. J. Watson Research Center, IBM, Yorktown Heights, NY, USA, in 2011, and at GlobalFoundries, Santa Clara, CA, USA, in 2014. She joined Intel Labs, Santa Clara, in 2015, as a Research Scientist. Her research interests include device-circuit co-design for low-power applications.

She was one of the winners of the IBM Ph.D. Fellowship Award from 2011 to 2012 Academic Year.

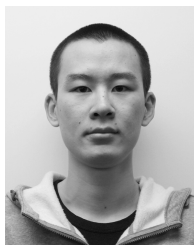


Hai (Helen) Li (M'08–SM'16) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. She was with Qualcomm Inc., San Diego, CA, USA, Intel Corporation, Santa Clara, CA, USA, Seagate Technology, Bloomington, MN, USA, the Polytechnic Institute of New York University, Brooklyn, NY, USA, and the University of Pittsburgh, Pittsburgh, PA, USA. She is currently a Clare Boothe Luce Associate

Professor with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. She has authored or co-authored over 200 technical papers published in peer-reviewed journals and conferences, and authored a book *Nonvolatile Memory Design: Magnetic, Resistive, and Phase Changing* (CRC Press, 2011). Her current research interests include memory design and architecture, neuromorphic architecture for brain-inspired computing systems, and cross-layer optimization for low power and high performance. She is a Distinguished Member of the ACM. She received the NSF CAREER Award in 2012, the DARPA Young Faculty Award in 2013, the TUM-IAS Hans Fisher Fellowship in 2017, seven best paper awards, and additional seven best paper nominations. She was a General Chair of ISVLSI, ICCE, ISQED, and GLSVLSI, and the Technical Program Chair of SoCC, iNIS, GLSVLSI. She serves as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, the IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the IEEE TRANSACTIONS ON EMBEDDED COMPUTING SYSTEMS, *IEEE Consumer Electronics Magazine*, the *ACM Transactions on Design Automation of Electronic Systems*, and *IET Cyber-Physical Systems: Theory and Applications*.



Jae-Sun Seo (M'10–SM'17) received the B.S. degree from Seoul National University in 2001, and the M.S. and Ph.D. degrees from the University of Michigan in 2006 and 2010, respectively, all in electrical engineering. He spent graduate research internships at the Intel Circuit Research Lab in 2006 and the Sun Microsystems VLSI Research Group in 2008. From 2010 to 2013, he was with the T. J. Watson Research Center, IBM, where he was involved in cognitive computing chips under the DARPA SyNAPSE Project and energy-efficient integrated circuits for high-performance processors. In 2014, he joined Arizona State University as an Assistant Professor with the School of ECEE. In 2015, he was a Visiting Faculty with the Intel Circuits Research Lab. His research interests include the efficient hardware design of machine learning/neuromorphic algorithms and integrated power management. He was a recipient of the Samsung Scholarship from 2004 to 2009, the IBM Outstanding Technical Achievement Award in 2012, and the NSF CAREER Award in 2017. He serves on the Technical Program Committee for ISLPED since 2013, ISOCC from 2016 to 2017, DAC in 2018, the Review Committee for ISCAS from 2017 to 2018, and on the Organizing Committee for ICCD from 2015 to 2017.



Chang Song received the B.S. degree in electronic and information science and technology from Peking University, Beijing, China, in 2015, and the M.S. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. His research interests include neuromorphic computing using memristors and the adversarial robustness of neural networks.