A Survey on Neuromorphic Computing: Models and Hardware

Amar Shrestha[1], Haowen Fang[1], Zaidao Mei[1], Daniel Patrick Rider[1], Qing Wu[2] ,Qinru Qiu[1]
[1]Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY
[2]US Air Force Research Laboratory, Rome, NY, USA
Email: [1]{amshrest, hfang02, zmei05, dprider, qiqiu}@syr.edu, [2] qing.wu.2@us.af.mil

***Abstract*—The explosion of "big data" applications imposes severe challenges of speed and scalability on traditional computer systems. As the performance of traditional Von Neumann machines is greatly hindered by the increasing performance gap between CPU and memory ("known as the memory wall"), neuromorphic computing systems have gained considerable attention. The biology-plausible computing paradigm carries out computing by emulating the charging/discharging process of neuron and synapse potential. The unique spike domain information encoding enables asynchronous event driven computation and communication, and hence has the potential for very high energy efficiency. This survey reviews computing models and hardware platforms of existing neuromorphic computing systems. Neuron and synapse models are first introduced, followed by the discussion on how they will affect hardware design. Case studies of several representative hardware platforms, including their architecture and software ecosystems, are further presented. Lastly we present several future research directions.**

***Index Terms*—Neuromorphic computing, spiking neural networks, bio-inspired computing, machine learning.**

I. Introduction

The ever-increasing scale and computation complexity of machine intelligence have been posing challenges on the traditional Von Neumann architecture and demanding for higher performance per watt efficiency from energy limited systems such as edge devices, Internet-of-Things (IOT), and cyber physical systems (CPS). This motivates a new paradigm of massively parallel and distributed computing inspired by biological neural systems, namely neuromorphic computing. By learning from the biological and physical characteristics of the neocortex system, researchers in neuromorphic computing incorporate a brain-inspired computing model, a non-conventional architecture, and novel device technology to provide energy efficient solutions to real-life machine intelligence problems.

The concept of neuromorphic computing was first proposed by Carver Mead in the 1980s [1]–[4]. The early works in this area focused on emulating the analog behavior of neural systems. It is observed that biological systems achieve many orders of magnitude higher efficiency than digital systems when performing certain cognitive tasks. [1] and [4] credit such advantage to the fundamental differences between digital circuits and biological systems. The early works in neuromorphic computing tried to bridge the gap between the lower-level physical details of biological systems and the higher-level computational functionality. [2], [5] claim that, due to their adaptability, neuromorphic systems are more resilient to noise and component failure and have the potential to be more energy efficient.

The early efforts of neuromorphic computing include [1]–[4], [6]–[9]. Those works mainly focus on modeling realistic biological systems using analog circuits. [7] developed a silicon retina and a sensorimotor system. [8] designed an electronic cochlea using CMOS which shares the same principle as biological cochlea. [1] proposed a chip that is structurally similar to retinas of higher animals. [10] developed a floating-gate silicon MOS transistor to emulate synapse and realized a learning rule on the synapse array.

The implementation of neuromorphic computing has shifted to the digital domain in recent decades for better noise resilience and higher scalability. The research focus has also extended from single neuron implementation to network and inter-neuron communication architectures. In addition to digital systems, emerging materials and devices such as memristors, phase changing materials, photonic circuits are also being investigated for hybrid solutions of neuromorphic computing. Spiking neural network (SNN) is often studied together with neuromorphic computing as the underlying computational model. Sometimes the two terms are even interchangeable. SNNs have more biologically plausible features than conventional artificial neural networks (ANNs) [11]. Similar to the biological neural system, SNN is inherently a dynamic and stateful network. The most distinct property of SNN is that the information is represented, transmitted and processed as discrete spike events, also referred to as action potentials [12]. Spikes are electrical pulses in biological neural systems. In SNN mathematical models, spikes are usually represented by Dirac Delta functions. Although a spike enables low power information transmission and processing, the non-differentiable Dirac Delta function also imposes a major challenge in SNN training, hindering the application of gradient descent algorithms [13]. In addition, unlike ANN, in which inter-neuron connections pass information lossless with a linear scaling controlled by the weight coefficients, connections/synapses of SNN may consist of multiple state variables and parameters. This feature makes the SNN more powerful in processing spatial/temporal sequences, but also increases the complexity of its implementation.

It is noteworthy that the boundary between SNN and ANN is not always clear. Though most SNN models use spikes, there are also rate-based SNN models, in which the output

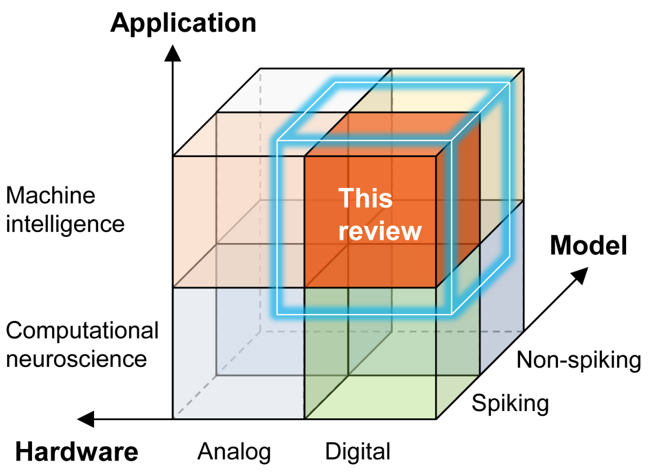


Fig. 1: Different aspects in neuromorphic computing.

of a neuron is no longer discrete spikes, but real-valued instantaneous spike rates. Such models can be interpreted as ANNs [14]. There are also models [15], [16] and hardware [17] that fuse SNN and ANN together. In this work, the name SNN is used to refer to the models that generate spikes as their outputs.

While the inferencing and learning of conventional ANNs are generally formulated as matrix-vector multiplications, there is no unified model for SNNs. Different models for spiking neurons and synapses represent their biological counterpart at different levels of details, which impacts the flexibility, complexity, and efficiency of hardware/software implementations. Based on their applications, we can divide neuromorphic computing into two categories, systems for computational neuroscience and systems for machine intelligence. Although their boundary is not always clear, the former usually focuses on models with more biophysical details and tries to reproduce their physiological features such as network oscillations. The latter focuses more on mathematically abstract models and their information representation and retrieval abilities. In Figure 1, we divide neuromorphic computing systems into 8 main categories based on their computational model, implementation, and applications. In this paper we will limit ourselves to the digital or mixed signal implementation of spiking neural networks for machine intelligence applications. Compared to earlier survey [18], which comprehensively discusses various aspects of neuromorphic computing, including history, model, algorithm, hardware design, device and applications, this work focuses more on the algorithm-hardware codesign. For example, we will discuss the implications that neuron models and learning algorithms may impose on hardware design, how the hardware architecture limits software and algorithm, and the design trade-offs between algorithm and hardware.

The rest of the survey is organized as the following. Section II reviews neuron and synapse models, network topologies, information encoding schemes and learning algorithms. Their impact on hardware implementation will be discussed in Section III, followed by a detailed discussion of the hardware and software ecosystems of several selected neuromorphic computing systems in Section IV. The outlook of future research directions will be given in Section V.

II. Neuromorphic Computing Models

Biological neurons communicate with each other by generating and propagating electrical pulses called spikes [19], [20]. At the high abstraction level, all spiking models share the following common properties: (1) they process information coming from many inputs and produce single or multiple spikes; (2) the probability of spike generation is increased by excitatory inputs and decreased by inhibitory inputs; (3) at least one state variable is used to characterize their dynamics and the model is supposed to generate one or more spikes when the internal variables of the model reach a certain state. Neurons connect and communicate with one another through specialized junctions called synapses [21], [22]. Similar to the neuron models, synapse models also vary in the complexity and biological plausibility.

The details of some popular spiking neuron models and synapse models are reviewed in Sections II-A and II-B. Different spike coding techniques are reviewed in Section II-C. In Section II-D, we discuss various network architectures and in Section II-E we show how learning is accomplished in the networks of spiking neurons.

A. Neuron Models in Ordinary Differential Equations (ODE)

The existing neuron models can be categorized into two groups, conductance-based models and spike-based models. The former includes the Hodgkin–Huxley (HH) model [23], the Fitz-Hugh-Nagumo (FHN) model [24] and the Morris-Lecar [25] model, while the latter includes the Izhikevich model [26], the Integrate and Fire (IF) model and the Leaky-Integrate and Fire (LIF) [27] model.

Conductance-based models are based on an equivalent circuit representation of a cell membrane, as first put forth by Hodgkin and Huxley [23]. These models apply a set of nonlinear differential equations to provide a biophysical interpretation of an excitable cell in which current flows across the membrane due to the charging of the membrane capacitance (I_c) and the movement of ions across ion channels (I_{ion}), such that the total membrane current $I_m(t)$ is the sum of the capacitive current and the ionic current $I_m(t) = I_c + I_{ion}$. The membrane potential V_m of the cell with capacitance C_m is related to the capacitance current based on the following equation

$$I_c = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} \tag{1}$$

The ion current I_{ion} is a function of the difference of the V_m and the ion potential, whose conduction is time varying and modeled by a set of differential equations. Based on the model, positive surges (i.e. spikes) are formed on the membrane potential at constant or time varying input current. The conductance-base models consider neuron input, output, and state as continuous-time continuous-valued variables; hence they have a high computational complexity. Due to their high fidelity to the biological neuron, the conductance-based models are more widely used in computational neuroscience.

The spike-based model simplifies the neuron input and output into spikes. A sequence of the spike events, i.e. a spike train, can be described as the following

$$S(t) = \sum_f \delta(t - t^f), \tag{2}$$

where $f = 1, 2, \cdots$ is the label of the spike and $\delta(.)$ is a Dirac function with $\delta(t) \neq 0$ for $t = 0$ and $\int_{-\infty}^{\infty} \delta(t)dt = 1$. The basic assumption underlying most spiking neuron models is that it is the timing of spikes rather than the specific shape of spikes that carries neural information [28].

Among the spike-based models, the *Integrate-and-Fire* (*IF*) model, and *Leaky Integrate-and-Fire* (*LIF*) model [28] are the most widely used. Both models abstract biological neurons as point dynamical systems. The dynamics of the LIF unit is described by the following formula:

$$C\frac{\mathrm{d}u(t)}{\mathrm{d}t} = -\frac{1}{R}u(t) + (i_o(t) + \sum w_j i_j(t)) \tag{3}$$

where $u(t)$ is the membrane potential, C is the membrane capacitance, R is the input resistance, i_o(t) is the external current driving the neural state, i_j(t) is the input current from the j-th synaptic input, and w_j represents the strength of the j-th synapse. Both $i_o(t)$ and $i_j(t)$ are functions of spike trains, as given in Equation 2. When $R \to \infty$, formula 3 is reduced to an IF model. In both IF and LIF models, a neuron is supposed to fire a spike, whenever the membrane potential u reaches a certain value v referred to as the firing threshold. Immediately after the spike, the neuron state is reset to a new value $u_{res} < v$ and holds at that level for the time interval representing the refractory period.

The majority of the neuromorphic systems utilize IF and LIF neurons as they are easier to implement and are computationally efficient. The LIF model has been extended with one or more adaptation variables to account for different firing patterns. A well-known model is the Izhikevich model, which can produce firing patterns experimentally verified on neocortical and thalamic neurons [26]. However, it is not clear what roles the different firing patterns are playing in learning and cognition, and those additional adaptation variables increase the model complexity. Therefore, they are less used in machine intelligence applications.

B. Neuron Dynamics in Spike Response Model (SRM)

The aforementioned IF and LIF models are over-simplified by considering the synaptic connection as a time-invariant device with a constant efficacy w and assume that the membrane potential reset as an instantaneous procedure. A more realistic neuron model considers the dynamics in the neuron and synapse behavior.

The arrival of a presynaptic spike triggers the synaptic electric current flowing into the biological neuron [20]. It causes a change in the membrane potential of the synapse, which is referred to as post-synaptic potential (PSP). In a general form, the time course of j^{th} PSP can be described as the convolution of the presynaptic spike train $S_j(t)$ and a kernel function $K_j(t)$ scaled by a weight coefficient w_j as the following:

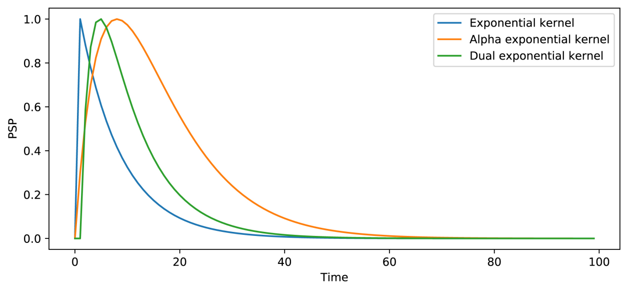


Fig. 2: Exponential, Alpha and dual exponential Kernels.

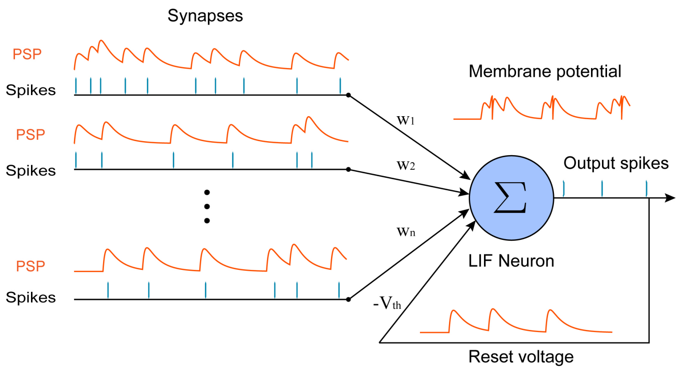


Fig. 3: Spike response model.

$$\begin{aligned} PSP_j(t) &= w_j \int_0^{\infty} S_j(t-s)K_j(s)\mathrm{d}s \\ &= w_j \sum_{t_j{}' < t} K_j(t - t_j{}'), \end{aligned} \tag{4}$$

where $t_j{}'$ is the time of spikes on the input $S_j(t)$. $K(t)$ can be an exponential, a dual exponential or an alpha kernel defined by following equations:

$$K(t) = e^{-\frac{t}{\tau}} \tag{5}$$

$$K(t) = \frac{t}{\tau}e^{-\frac{t}{\tau}} \tag{6}$$

$$K(t) = V_0(e^{-\frac{t}{\tau_m}} - e^{-\frac{t}{\tau_s}}) \tag{7}$$

Their spike responses are illustrated in Figure 2.

The reset of the membrane potential is no longer instantaneous. Instead, it is modeled as a negative potential induced by the output spike train $S_o(t)$ going through a kernel function $h(t)$,

$$R(t) = \int_0^{\infty} S_o(t-s)h(s)\mathrm{d}s = \sum_{t_o{}' < t} h(t - t_o{}'), \tag{8}$$

where $t_o{}'$ is the time of spikes on the output spike train. Usually $h(t)$ is a kernel given in Equation 5.

The way to interpret the neuron dynamics is as a convolution of the impulse response of a filter with the input spike train as

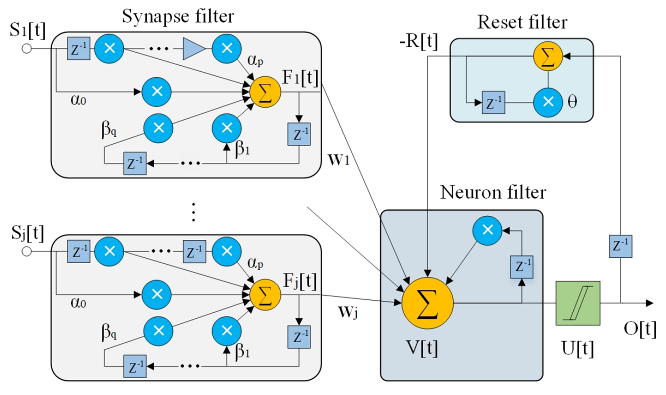


Fig. 4: Neuron modeled by digital filters [29].

in Equations 4 and 8, and is referred to as the Spike Response Model (SRM).

The membrane potential is the combined effect of $PSP(t)$ and $R(t)$ as shown in Figure 3. Using SRM representation, it can be represented as an integral over the past input and kernel responses. A typical SRM model is defined as the following [12]:

$$V_m(t) = \sum_{j=1}^{N} w_j \sum_{t_j^{'}<t} K(t - t_j^{'}) - V_{th} \sum_{t_o^{'}} h(t - t_o^{'}) \quad (9)$$

where $V_m(t)$ is the membrane potential, $K(t)$ and $h(t)$ are two convolution kernels associated to synaptic dynamics and membrane potential reset events. When $V_m(t)$ exceeds the threshold V_{th}, the neuron generates a spike output whose time is indicated by the spike time $t_o^{'}$.

By using a kernel $K(t)$ with arbitrary shape, the SRM model provides complicated dynamics and rich temporal information. The simplified LIF model in Equation 3 is a special case of the SRM model, where the $K(t)$ and $h(t)$ are two low-pass filters. The SRM model shows that the membrane potential is a function based on not only the current but also the past input spikes, which explains the neuron's ability to respond to temporal patterns.

The kernels in the SRM model can be implemented as discretized digital filters. Using the Z-transform [30], [31], they can be represented as a Linear Constant-Coefficient Difference (LCCD) equation in the following form:

$$y\,[t] = \sum_{p=1}^{P} \alpha_p y[t-p] + \sum_{q=1}^{Q} \beta_q x[t-p], \quad (10)$$

where $y[t]$ and $x[t]$ are the output and input of the kernel and P and Q are the feedback and feedforward orders. Using this implementation, a neuron in the SRM model can be represented as a network of IIR filters, as shown in Figure 4. This architecture was adopted in [32]–[34] for the digital implementation of SRM neurons.

C. Neural coding and spike timing

Neural coding is an essential part of the SNN. It refers to the way in which information is represented by discrete spikes. Neural coding is tightly coupled with the neuron model and determines the performance of the SNN and hardware implementation.

Exactly how the brain and sensory system encode information is not fully understood yet. Rate coding and temporal coding are two commonly used information coding in neuromorphic computing. Rate coding represents a value by the number of spikes in a unit time. It agrees with the observation that the sensory nerves' spike frequency increases as the stimulus intensity increases. Rate coding has been widely adopted. For example, most SNN models and neuromorphic hardware for image classification use rate coding, where the pixel value is represented by the number of spikes in unit time [13], [35]–[37]. However, rate coding has its limitations. First of all, it introduces latency. The firing rate cannot be determined accurately until a sufficiently large number of spikes have been received. While a typical neuron firing rate is between 1 and 200 Hz, in realistic biological neural networks, there is not enough time to integrate spikes to get the spike count. For example, a fly can respond to a visual object after one or two spikes are received [12]. Secondly, rate coding is not energy efficient. It represents large values using high spike frequency, which increases the switching activities in computing hardware, and may even pose challenges to neuromorphic chip design [38]–[40]. Without extended latency or escalated spiking frequency, rate coding will suffer from high quantization error. When spikes are generated as stochastic events, there will be sampling errors too.

Temporal coding takes spike timing into account [41]–[43] such that the temporal structure of a spike train can convey information. Two spike-trains with the same spike count could represent distinct information, as shown by [44], and produce significantly different postsynaptic current. When considering the spike timing, the information capacity of a spike train is significantly increased [45]. However, temporal coding is still not well understood. There are many hypotheses, which lead to different variations of temporal coding schemes. For example, [46] shows that the spatial structure of an image is encoded by retinal ganglia using the relative timing of first spikes, referred to as latency coding. Latency coding assumes that the first spike carries the most significant information, while the subsequent spikes are less important. The latency coding is also known as Time-to-first-spike (TTFS) coding [12]. It is noteworthy that there are some subtle differences between the latency coding observed in a biological system and the TTFS in the context of neuromorphic computing. The latter utilizes at most one spike per neuron to encode information by applying a long refractory period or a strong inhibition [47], [48], while there is no such restriction in biological systems. [49] proposes reverse coding, which assumes that a stronger stimulus is encoded by a later spike time. This can be interpreted as a variation of TTFS. Training algorithms and neuromorphic hardware have been designed specifically for TTFS coding [47]–[50]. TTFS usually allows more efficient hardware because it substantially reduces spike numbers, hence the communication workload is less. Furthermore, neurons using TTFS do not have to accumulate multiple spikes to produce output, hence the computation latency is also reduced. As another variation of temporal coding, phase coding considers the entire spike train. Information is represented by the relative spike timing with respect to periodical background oscillation [12].

[51] and [52] suggest that different coding schemes may co-exist in the nervous system, and the brain uses different coding for different tasks. The variety and task specialization of coding schemes can also be seen in existing research in SNN. For example, [53], [54] encode image as spatial spike patterns. [55], [56] proposed to convert audio signals into time-varying spike patterns.

The choice of neural coding scheme is closely related to the decision on SNN training algorithms, neuron models and even the hardware architecture. For example, to recognize different temporal spike patterns, [57] employs LIF neuron with dual-exponential synapse defined in Equation 7. Every individual input spike builds up a time-varying PSP, which represents certain characteristics; [50] designs a dedicated single-spiking MAC circuit to support TTFS.

To utilize the information embedded in spike timing, neuron models with certain temporal dynamics, as discussed in Section II-B, must be used. For example, [33], [57]–[59] use SRM or its variants to learn spike timings. However, these models are not readily supported by some of the existing neuromorphic hardware. For example, TrueNorth uses a simplified LIF model, where a neuron's membrane potential is the accumulation of weighted input spikes with a constant leakage. While this architecture provides extremely high neuron/synapse density and energy efficiency, it is not suitable to implement the aforementioned models of temporal coding. A few other neuromorphic systems have memory allocated to each synapse to store the temporal dynamics. For example, Loihi allows a synapse to have three different state variables, which can be configured as traces. [34] reported an FPGA based SNN where each neuron core has a dedicated memory bank for the post-synaptic potential.

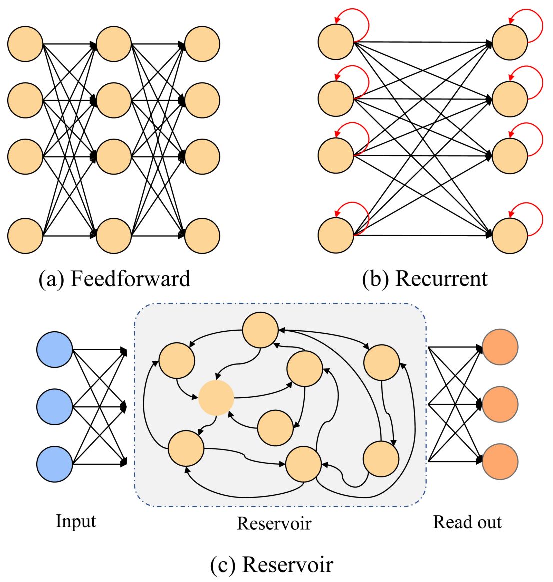


Fig. 5: ANN-inspired topologies.

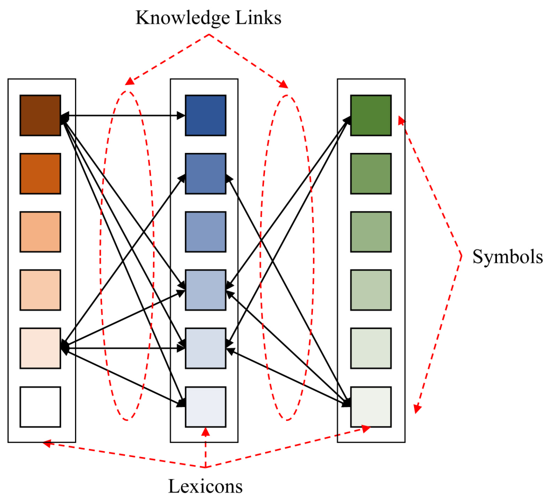


Fig. 6: Confabulation network.

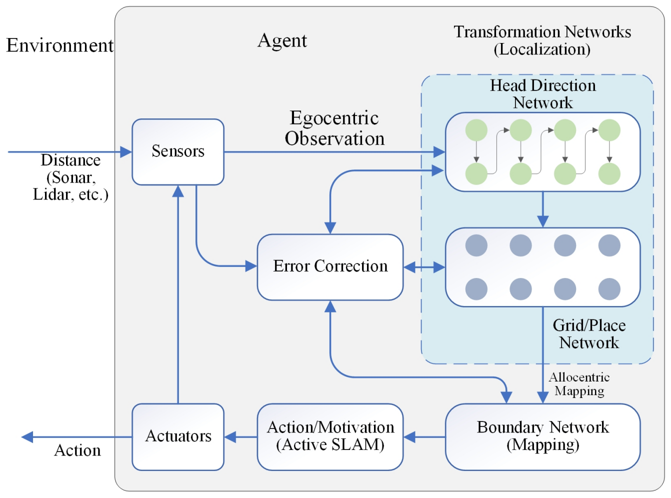
D. Network Topologies

Given the models of neurons and synapses, a spiking neural network can be constructed. Based on the network topology, we divide the SNNs into three categories, feedforward, recurrent and bio-inspired networks. Feedforward and recurrent networks are inspired by ANNs as shown in Figure 5 whereas the bio-inspired networks mimic the structure of various biological neural motifs.

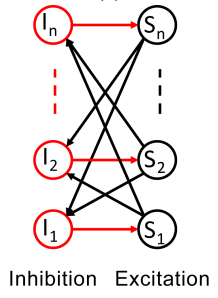
1) ANN-inspired: Feedforward network is the simplest form of neural networks where the information moves in only one direction, from the input layer, through the hidden layers and to the output layer.

The recurrent SNN can be further divided into two categories, recurrent with synchronization and recurrent without synchronization. Recurrent structure is widely used in ANNs to detect temporal patterns in input sequences or generate temporally correlated outputs. In these ANNs, the hidden state of the network induced by previous input loops back to be processed with the current input to generate new hidden states and outputs. The synchronization between the hidden state of the previous cycle and the input of the current cycle is difficult to achieve in SNN, due to the inherent asynchronous and even driven nature of the SNN neurons. Because the input and hidden state variables are represented by a sequence of spike trains, special store-and-release neurons must be used to gate and release the spike sequences in order to synchronize them [60], [61]. It was not until recently that some works presented techniques on translating recurrent networks to SNNs [60], [61] or introducing dynamics into the neuron and synapse models to implement a form of recurrence [62]. We refer to these networks as recurrent SNNs with synchronization.

A large set of neural networks use the recurrent structure to stabilize signals and suppress noises. No synchronization among neurons is attempted in these networks. The feed-

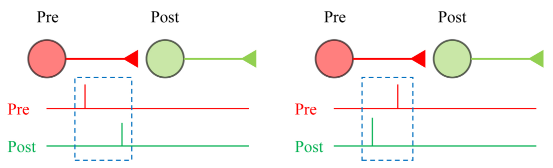


(a)

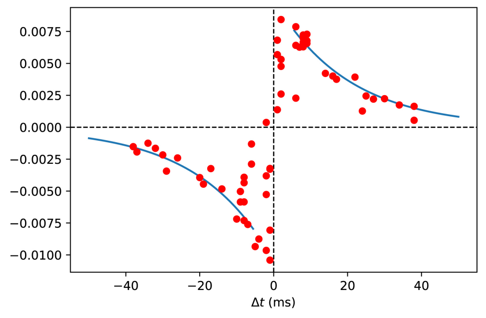


(b) Winner-Take-All network

Fig. 7: Bio-inspired topologies.



(a)



(b)

Fig. 8: (a) STDP and (b) its profile.

back and the input eventually reach a dynamic equilibrium, which defines the network state. For example, Echo State Network (ESN), also referred to as reservoir network, is a variation of recurrent networks which consists of a hidden layer (reservoir), containing neurons randomly connected to each other with fixed weights, and connected to the output layer which feeds back to the reservoir with plastic weights [63]–[69] as shown in Figure 5(c). In a reservoir network, the output layer is used to classify the state of the network. Learning takes place only in the output layer, which consists of conventional (i.e. non-spiking) neurons. Another example is the spiking confabulation network. Cogent confabulation is a connection-based cognitive model that captures correlations among features at the symbolic level, as shown in Figure 6. It describes the basic dimensions of the observation using a set of features referred to as lexicons. The attributes of a given feature are referred to as the symbols, which are analogous to neurons in the biological nervous system. Their pairwise conditional probabilities are referred to as the knowledge links. When implemented with Bayesian spiking neurons as in [70], [71], the neurons interact with each other and the equilibrium state of the spiking rates infers the likelihood of the symbols represented by each neuron.

A large number of SNNs are arranged to follow biological neural architectures. For example, simultaneous localization and mapping (SLAM) [72] networks get inspiration from the navigation system in the hippocampus and entorhinal cortex of rats, where different types of spatially-tuned neurons were found: the head-direction cells are sensitive to the heading direction of the animal, place cells are active each time the animal visits a particular part of the environment, and grid cells presumably perform path integration [73]. As another example, Winner-Take-All (WTA) networks containing recurrent connectivity between inhibitory and excitatory neurons are common models to explain decision-making and action selection in the cortex [74], [75]. They are widely used for unsupervised learning and feature selection in SNNs [76], [77]. Figures 7a and 7b illustrate the structure of the two aforementioned networks.

The hierarchies in the sensory cortex are of particular interests to research in sensor signal processing. The mammalian olfactory system contains three major hierarchical levels including the epithelium where the stimulus enters the nervous system, the olfactory bulb (OB) where the first transformation happens, and the piriform cortex (PC) which integrates and stores the information relevant for odor recognition [78], [79]. Such hierarchical network structures have been used for recognition and decision-making tasks in SNNs [80]–[82]. Medial Superior Olive (MSO) in the mammalian auditory pathway is responsible for sound localization. They are organized spatially as a place map of location [83], [84] and at a higher level these subgroups are subsequently also organized into frequency selective clusters. These MSO systems inspired [85], [86] to utilize SNNs for sound classification and localization. Attractor networks, whose activity tend towards dynamic stability, have been posited to help explain eye control, working memory, head direction, locomotion and olfaction [87] and thus have been utilized for such control tasks in SNNs [88]–[90].

E. Learning

The ability of synaptic connections to change their efficacy is referred to as synaptic plasticity. This is thought to be the basic mechanism underlying learning and memory in biological neural networks [91]. Various forms of synaptic plasticity co-exist. Some are determined only by the history of presynaptic stimulation, independently of the postsynaptic responses [92]–[94]. Others depend on the temporal order of pre- and postsynaptic activities [92], [95], [96].

In general, synaptic potentiation (i.e. the increase of synaptic efficacy) is observed when presynaptic spikes precede postsynaptic spikes, as it indicates a causal relationship. The reversed order of spikes induces synaptic depression (i.e. the decrease of synaptic efficacy). This phenomenon is called Spike-Timing-Dependent-Plasticity (STDP). It can be used for unsupervised learning. A popular choice for the STDP rule [97] for potentiation Δw_+ and depression Δw_- is given as:

$$\begin{aligned} \Delta w_+ &= +A_+ \exp\left(-\Delta t/\tau_+\right) \quad \text{for } \Delta t > 0 \\ \Delta w_- &= -A_- \exp\left(+\Delta t/\tau_-\right) \quad \text{for } \Delta t < 0 \end{aligned} \tag{11}$$

where τ_+ and τ_- determine the ranges of pre- to postsynaptic interspike intervals over which synaptic strengthening and weakening occur. A_+ and A_- determine the maximum amounts of synaptic modification, and Δt is the time of the postsynaptic spike minus the time of the presynaptic spike.

A wide range of SNN algorithms that can learn temporal spike patterns employ more biologically realistic LIF neuron models with alpha or dual-exponential synapse [57], [77], [98]–[100] as shown in Section II-B. In these models, the PSP decays exponentially over time, hence, can be utilized as a metric to reflect temporal dependency. This type of neuron does not simply accumulate weighted spikes as membrane potential, instead, it integrates weighted time varying PSP, hence exhibiting complex temporal dynamic behavior. Various STDP learning rules have been proposed that update the weight based on different traces. Traces are decaying state variables reflecting the temporal history of input and output spikes. They correspond to the ion concentrations in a biological neuron. The PSP is an example of a trace variable. A pairwise STDP rule using traces [101] is given as:

$$\Delta w = A_+ x(t) S_x(t) - A_- y(t) S_y(t) \tag{12}$$

where $x(t)$ and $y(t)$ are the pre- and postsynaptic traces. $S_x(t) = \sum_x \delta(t - t^x)$ and $S_y(t) = \sum_y \delta(t - t^y)$ are the pre- and postsynaptic spike trains. Thus, from Equation 12 and Figure 8a, the weight is increased at the moment of postsynaptic firing by an amount that depends on the value of the trace $x(t)$ left by the presynaptic spike. Similarly, the weight is depressed at the moment of presynaptic spikes by an amount proportional to the trace $y(t)$ left by previous postsynaptic spikes. This has been shown to fit the experimental data as shown in Figure 8b and [95], and has been studied in [12], [102], [103].

The most straightforward way to implement supervised learning is to use Hebbian Learning [77], [104]. Supervision is introduced in Hebbian learning by an additional 'teaching' signal that reinforces the postsynaptic neuron to fire at the target times and to remain silent at other times. The 'teaching' signal is usually transmitted to the neuron in a form of synaptic currents or as intracellularly injected currents.

Another approach is to utilize a supervised learning algorithm for ANNs called backpropagation. Backpropagation, a gradient-based optimization algorithm, is a standard training technique for ANNs. However, it cannot be directly applied to the in-hardware learning of an SNN running on a neuromorphic processor due to several reasons; (1) spiking neuron's activities are not differentiable, (2) the connections between neurons in SNNs are unidirectional such that a backward path must be added explicitly with constantly updated weights during learning, (3) errors in ANNs are propagated as real values and (4) weight update of a synapse is not solely dependent on locally available information as required in a neuromorphic hardware [105]. There have been various approaches to adopt the backpropagation algorithm to train deep SNNs directly [13], [33], [36], [106]–[108]. One category of approaches keeps track of the membrane potential at spike times and back-propagate errors based on that. SpikeProp [109] is the first attempt to train an SNN using such an approach. But SpikeProp is limited to single-spike learning. A similar category of approaches [110] [13] treats the discontinuities during spike times as noise and smoothens the membrane potential to essentially make it continuous. These approaches utilize spike-rate to compute the loss and membrane potential to compute the error derivative, and hence create a discrepancy. [106] proposed an event-driven random backpropagation (eRBP) algorithm simplifying the backpropagation chain path. But this work requires multicompartmental neurons to enable error to locally modulate plasticity. In [107], a supervised learning method was proposed (BP-STDP) where the backpropagation update rules were converted to temporally local STDP rules for multilayer SNNs. Recently, Error-Modulated STDP (EM-STDP) [108], [111] was proposed to approximate backpropagation in the spike domain for neuromorphic implementation. This work applies the same type of integrate and fire (IF) neuron in the forward and backward path, and enhances the biological plausibility of backpropagation algorithm by introducing a weight update rule that resembles the rate-based STDP using only the locally available information. Its learning capability has been demonstrated on the Loihi processor [111].

III. Neuromorphic System Design Considerations

To design a neuromorphic processor, a complete ecosystem including both software and hardware needs to be considered. This does not only include the hardware implementation of neurons and synapses, and their communication network, but also simulators and compilers for design validation and optimization.

A. Neuron and Synapse Implementations

A bottom-up approach is generally adopted in neuromorphic hardware design. Neurons and synapses are the building blocks, whose implementations are designed first. These building blocks are connected by a communication network to form a system architecture. Different implementations and hardware architectures are selected for neuron and synapse models with different degrees of complexity. The leaky integrate and fire (LIF) model in Equation 3 has been a popular choice for hardware implementation [105], [112]–[114] since it is simple but still retains some temporal dynamics. Complex neuron and synapse behavior specified in Equation 9 requires specific hardware to efficiently compute their evolution through time. The LIF can be reduced into an IF model, which can be implemented cost effectively using an adder, a comparator, and a memory [11], for input integration, threshold detection and membrane potential storage, respectively.

To achieve higher degree of fidelity to biological models, ionic channels and other bio-realistic components have been implemented [113], [115], [116]. [117] implements advanced reconfigurable units based on the work of Izhikevich [26] or bio-realistic ion channels [116] interaction in fully digital designs. The SpiNNaker [118] can be used to evaluate detailed biological neuron and synapse models at a high computation cost. These implementations of highly bio-plausible neurons and synapses provide insights of the brain function from the neuroscience point of view. They usually are not used for machine intelligent applications.

B. Implementation Choices

Based on their implementation choices, neuromorphic systems can be categorized into three categories, (1) digital, (2) analog, or (3) mixed signal platform.

Digital neuromorphic systems can further be divided into CPU based, Application Specific Integrated Circuit (ASIC) based and FPGA based implementations. An example of CPU based implementation is SpiNNaker. SpiNNaker is an ARM based, fully digital massively parallel system. It is composed of thousands of ARM cores and a custom interconnect communication scheme optimized for spike-based network communication. The processing unit itself is general purpose and not customized for neuromorphic functions [40], [118]–[133].

IBM's TrueNorth [114], [117], [134]–[139] and Intel's Loihi [105] are well known examples of fully custom ASIC implementation of neuromorphic systems. Some other examples of ASIC based neuromorphic systems include [63], [64], [140]–[155].

Most ASICs are subject to limitations of specific neuron models and algorithms. Therefore, FPGA has also drawn much attention for its flexibility. FPGA has been widely used for exploring various aspects of neuromorphic hardware and algorithms research. Most of these works adopt a multi-core architecture [34], [156]–[159]. Due to the limited resource of a single FPGA, there are also works utilizing multiple FPGAs [160]–[162]. FPGA's flexibility also lends it for exploration into various in-hardware training algorithms. Some examples are: a modified STDP rule that uses shift operation to replace the exponential operation to reduce logic resource consumption [157]; competitive Hebbian learning on chip with biologically plausible Izhikevich neurons on FPGA [163]; a hardware friendly STDP rule which allows low bit precision in a liquid state machine (LSM) on FPGA [164]; STDP for convolutional SNN on FPGA [165]; and an STDP rule that uses only 1-bit synaptic weights to reduce computing, communication, and memory overhead [166].

For different biological neuron behaviors, such as conservation of charge, amplification, thresholding and integration, the analog circuit analogies can be found [2]. Such similarity makes analog integrated circuits and neuromorphic systems well suited for each other. The original neuromorphic definition by Carver Mead referred to analog circuits that operated in subthreshold mode [2]. Many analog neuromorphic systems also operate in this region typically for power efficiency [90], [167]–[176]. There are a large variety of other neuromorphic analog implementations [90], [177]–[209].

Similar to digital FPGAs, there are field programmable analog arrays (FPAAs) enabling programmability for analog neuromorphic systems [210]–[214]. Some custom FPAAs are developed specifically for neuromorphic systems, including the field programmable neural array (FPNA) [215] and the NeuroFPAA [216]. While many of the digital neuromorphic systems adopt asynchronous and event driven methods for energy efficiency, analog neuromorphic systems do sometimes employ clocks for synchronization.

Mixed analog and digital implementation is usually the solution to overcome some inherent limitations of analog implementation. In many analog neuromorphic systems, synaptic weights are stored in digital memory for reliability and longer duration [217]–[220]. In some analog neuromorphic systems, digital communication is utilized either within the chip, or among neuromorphic chips [221]. These communications are usually in the form of digital spikes. Using digital components for programmability or learning mechanisms is also common [222]–[225]. Two major projects within the mixed analog/digital family are Neurogrid and BrainScaleS.

C. Architecture

In this section, we discuss three different architecture choices, their pros and cons, as well as implications on hardware design. These three choices are von Neumann architecture, ideal architecture for neuromorphic computing and practical mult-core architecture.

von Neumann Architecture Von Neumann architecture is the foundation of modern general-purpose computers. It is shown in Figure 9a . A typical von Neumann architecture consists of following components: a central processing unit (CPU), memories, and input/output (I/O) devices. Devices are connected through bus systems. Data and program are stored in external memory, and fetched by CPU sequentially. Such architecture sufferers from the well-known von Neumann bottleneck, i.e. the system performance is bounded by the data exchange speed between CPU and the external memory. Though von Neumann architecture provides high flexibility, it is not suitable for neuromorphic computing because it cannot provide the massive concurrency and parallelism featured in the biological neural systems.

Ideal Architecture In a biological system, each neuron and synapse has its own state, which can be characterized by a set of parameters and variables in software/algorithm models. These parameters/variable are not shared among different neurons or synapses, and they are updated locally and concurrently. The inter-neuron communication is also a parallel process through massive number of synapses. Based on above observations, an ideal architecture should support

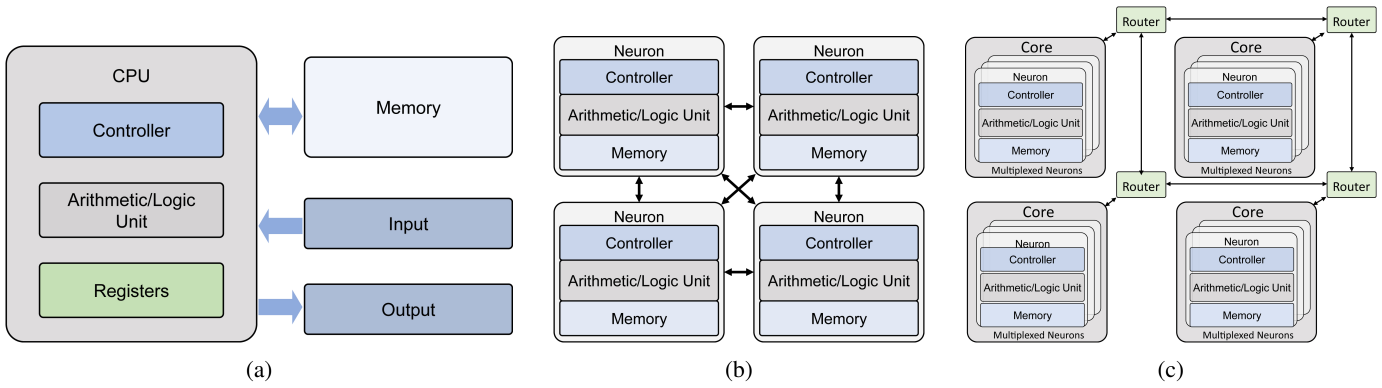


Fig. 9: (a) Von Neumann Architecture (b) Ideal Neuromorphic Architecture (c) Practical Neuromorphic Core.

TABLE I: Trade-offs of different architectures.

Architecture	Scability	Cost	performance
Von Neumann	++	++	—
Ideal	-	–	+++
Practical	++	+	++

1) Local and dedicated data storage; 2) Massive concurrency; and 3) High connection density.

Based on above requirements, an ideal architecture of digital neuromorphic hardware is presented in Figure 9b where each processing unit and its local memory are used to represent a single neuron, and the local Arithmetic/logic unit (ALU) is responsible for updating neuron status. The close-to-memory computing reduces data retrieving latency, while the distributed memory enables parallel computation. This ideal architecture maximizes the number of synaptic operations per second (SynOps/s), which is an important measure of neuromorphic hardware performance.

Practical Architecture The aforementioned ideal neuromorphic architecture, that maintains one processing unit for each neuron, is not scalable when the size of the neural network increases. The large circuit overhead arising from an ALU assigned to each neuron and hardwiring the neurons to each other for large-scale SNNs is highly impractical. A practical solution is to group a number of neurons in a core, as shown in Figure 9c. These neurons have their own local data, but share the same data path to update neuron and synapse status. Compared to the ideal neuromorphic architecture, this reduces the effective circuit area per neuron significantly. The cores also enable sharing of common parameters among the neurons for a more efficient usage of memory. However, as the same ALU is utilized to update the neurons and synapse status associated with a core, usually time-multiplexing is utilized. This reduces the parallelism, as the ALU can only be accessed by one neuron at a time. Additionally, this also introduces spiking delays (or delay in neuron update) that can cause errors in the neuronal encoding. Trade-offs of above three architectures are shown in Table I.

D. *Communication*

Neuromorphic systems support both intra-chip and inter-chip communication. Both types of communications are implemented using address event representation (AER). [226]–[228] apply AER to on-chip inter-neuron packet based communications. Vainbrand and Ginosaur studied different network-on-chip architectures for neural networks, including mesh, shared bus, tree, and point-to-point. They found network-on-chip multicast to have the highest performance [229]. Ring-based communication structure has been tested successfully [230], [231] for on-chip neuron communications. Buses were also utilized for some on-chip communication systems [232], [233]. This asynchronous bundled data design style is well suited for SNNs that fundamentally feature a high degree of sparseness in their activity across both space and time. [234]–[239] also apply AER to inter-chip communications, where the chip ID is encoded as part of the packet address. AER have been implemented through custom PCI boards to optimize performance [240], [241] or utilizing FPGAs [242]–[244].

E. *Supporting Software and Ecosystems*

Supporting software tools are important components in the ecosystem of neuromorphic processors. Those usually consists of tools for mapping, programming and simulation. The mapping tools partition an SNN into clusters and map clusters to processing units on the neuromorphic hardware [132], [245]–[250]. The goal is to minimize the inter-core communication that considers the hardware constraints such as the number of input/output channels, the amount of local memories, etc. Programming tools enable users to explicitly describe a particular neuromorphic architecture [251]–[255] by setting different parameters and topology configurations, or by utilizing custom training methods. Software simulators [131], [249], [253], [256]–[259] are used to emulate the neuromorphic hardware and enable the user base in developing and testing of network topologies, training algorithms, neuron parameters, etc., when the hardware has not been widely deployed.

IV. Case Studies of some Large-Scale Neuromorphic Systems

In this section, we will discuss several representative systems as examples to showcase the main components discussed in previous section. They are: neuromorphic super computing platform (SpiNNaker), digital ASIC (TrueNorth), digital ASIC with on-chip learning (Loihi), analog and mixed-signal design (BrainScaleS), and ANN-SNN hybrid design (Tianjic). A quick summary of the neuron and synapse models that they support, their implementation choices, architecture and software support is provided in Table II.

A. Digital ASIC: TrueNorth

TrueNorth is a brain-inspired digital chip with an interconnected network of 64x64 neurosynaptic cores, where each core has 256 incoming axons, a 256x256 synapse crossbar, and 256 neurons [138] as shown in Figure 10a. In total, there are 1 million spiking neurons and 256 million synapses in a TrueNorth chip. Binary synapses, with programmable weights, gate the information flow from axons to neurons. Each axon fans out to all neurons in a core in parallel, thus, providing a 256-fold reduction in communication volume in comparison to a point-to-point approach. TrueNorth implements these intra-core connections through SRAM crossbar memory whereas inter-core connections are implemented through spike-based message-passing network. Programmability of TrueNorth in terms of neuron parameters, synaptic crossbar connections, and inter-core connectivity allows a wide range of structures, dynamics, and behaviors.

1) Neuron and Synapse Implementation: TrueNorth's neuron model is based on the classic leaky integrate-and-fire neuron (LIF) with five basic operations: synaptic integration, leak integration, threshold comparison, spike generation, and membrane potential reset. The membrane potential $V_j(t)$ of the neuron j is updated according to these five operations as summarized in Equations 13, 14 and 15.

Synaptic integration:

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{N-1} S_i(t)x_i \tag{13}$$

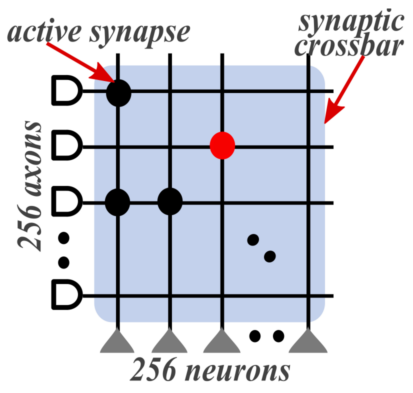
Leak integration:

$$V_j(t) = V_j(t) - \lambda_j \tag{14}$$

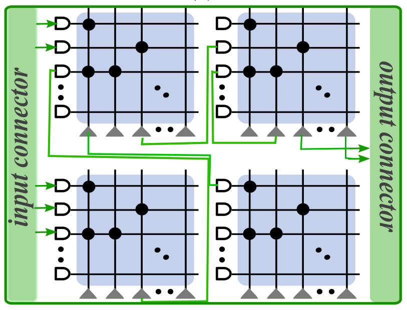
Membrane potential reset:

$$\text{If} \quad V_j(t) \geq \alpha_j, \text{ spike and } V_j(t) = R_j \tag{15}$$

For each of the neurons, membrane potential is the accumulated sum of the product of spike input to the synapse $S_i(t)$ at the current timestep and the signed synaptic weight x_i. Following integration, the LIF neuron model subtracts the leak value λ_j from the membrane potential every timestep. This linear leak operation serves as a constant bias on the neural dynamics. The neuron fires a spike and resets its membrane potential to R_j (typically, R_j is zero), when the membrane potential of the LIF neuron at the current timestep $V_j(t)$ is greater than or equal to the neuron threshold voltage α_j.



(a)



(b)

Fig. 10: Truenorth architecture (a) Functional view (b) A corelet [138].

This LIF neuron model is augmented by configurable and reproducible stochasticity [117]. Each individual neuron can be configured to have stochastic synaptic input, leak, and threshold to enable rich dynamics across population and time. The neuron model allows for four leak modes that bias the internal state dynamics in four different ways so that neurons can have radically different responses to identical inputs. The leaks can be either positive or negative to let the membrane potential to diverge from or converge towards a resting potential. The neuron model also provides two types of threshold; deterministic and stochastic, so that neurons can fire at different patterns even with the same accumulated membrane potential. It has six reset modes to determine the membrane potential after firing, enabling a rich finite-state transition behavior. To reduce complexity, it adopts fixed-point arithmetic and the neuron model uses only simple addition and multiplexing arithmetic/logic units instead of complex function units such as multiplication, division, and exponentiation.

The synapses themselves are binary (1: connected, 0: disconnected). Each synapse connected to a neuron in the crossbar is allocated a choice of four 8-bit signed weights. The synapses of TrueNorth neurons are non-plastic, i.e. the synaptic weight cannot be modified during the runtime.

Exploiting the provided configurability, users can use TrueNorth neurons to implement a wide variety of computational functions, including arithmetic, control, data generation, logic, memory, classic neuron behaviors, signal processing, and probabilistic computations. The programmable leakage and threshold give neurons the capacity to support a variety of neural codes including rate, population, binary, and time-to-spike coding. Rich and diverse array of complex computations and behaviors can also be synthesized by composing multiple neurons together. For example, the 20 behaviors of the Izhikevich dynamical neuron model can be qualitatively replicated, using a small number of elementary neurons.

A TrueNorth chip, built in Samsung's 28-nm process technology, occupies 4.3 cm^2 area and contains 5.4 billion transistors. Each core has 104,448 bits of local memory to store synapse states, neuron states and parameters , destination addresses, and axonal delays. In total, the TrueNorth has 428 million bits of on-chip memory. Addititionally, TrueNorth's power density is 20 mW per cm^2 which is highly efficient in comparison to typical CPU's 50 to 100 w per cm^2.

The very high energy efficiency of the TrueNorth processor does not only come from the low-cost hardware and simplified function, but also from its mixed synchronous-asynchronous neuron architecture, which reduces the neuron switching activities by 99%.The average firing frequency of TrueNorth neurons is approximately 20 Hz, which is close to the frequency of the Beta Wave associated to normal waking consciousness. This activity is very sparse compared to the speed of modern silicon. Joined with extensive power-gating, event-driven computing and asynchronous communication, the sparse activity significantly improves the energy efficiency.

2) Architecture and Communication: Multiple neurosynaptic cores are connected using distributed on- and off-chip connectivity to construct complex networks. There is no global clock other than a 1-kHz global synchronization signal, which discretizes the neuron dynamics into 1-ms time steps and ensures one-to-one equivalence between software and hardware.

A two-dimensional mesh network of routers form the backbone for interconnecting the 64x64 core array. Each of the routers have five ports (north, south, east, west, and local) and communicates spike event between cores in a time-multiplexed manner. With this mesh network, a neuron can talk to an axon on any core. When a neuron spikes in a core, it looks up an axonal delay (4 bits) and the destination address (8-bit absolute address for the target axon and two 9-bit relative addresses representing core hops in each dimension to the target core) in the local memory and encodes it in a packet. This packet is injected into the mesh, where it hops from core to core - first in the x dimension then in the y dimension (deadlock-free dimension-order routing). The asynchronous router delivers spikes at 0.3fJ per bit per μm. A merge-split structure is used at the four edges of the mesh to serialize exiting spikes and deserialize entering spikes. This enables scaling the two-dimensional mesh across chip boundaries as tiles, similar to the mammalian neocortex.

The architecture is efficient because neurons that form a cluster can be mapped to the same neuron core and communicate using local connections. The remaining inter-core connection is sparse, which reduces the communication cost. Additionally, each spike event addresses a pool of neurons on a target core, reducing the number of long-range spike events. The tile based architecture also increases the fault tolerance, as the system usability is not disrupted by occasional defects at the core and chip level. Also, the architecture is flexible as each neuron is individually configurable, each synapse can be turned on or off individually, and the neurons and synapses support programmed stochastic behavior. Thus, the neuron model [117] supports a wide variety of computational functions and biologically relevant spiking behaviors. One of the limitations of the architecture is that, to reduce the hardware cost, each column in the crossbar supports only 4 different synaptic weights, ranked from 1-4; and all synaptic connections in the same row must select the weight in the same rank in the corresponding column.

3) Supporting Software/Software Ecosystem: A TrueNorth program is a complete specification on the connectivity and configurations of a network of neurosynaptic cores, along with its external inputs and outputs. The "corelet", abstraction is used to represent a TrueNorth program by only exposing external inputs and outputs while encapsulating all other details of the network of neurosynaptic cores as shown in Fig. 10b. The object-oriented Corelet Language is developed by IBM for creating, composing, and decomposing corelets. As part of the TrueNorth ecosystem, a Corelet Library that provides a repository of reusable corelets macros, and an end-to-end Corelet Laboratory that is a programming environment integrated with the TrueNorth architectural simulator are also provided [258]. In 2016, IBM released the Eedn deep learning framework [36] to facilitate the training and mapping of deep spiking neural network on the TrueNorth system.

Eedn-trained CNNs have matched state-of-the-art accuracy on benchmarks that previously required floating-point precision and unconstrained connectivity, while achieving throughput of 1,200-2,600 classifications on CIFAR dataset per second and power consumption of only 25-275 mW [36] on 2000 to 4000 cores. TrueNorth systems have been applied to real time handwritten character recognition and confabulation [70], anomaly detection [260], optical flow [261], unconstrained optimization [262], decoding EEG [263], medical image segmentation [264], etc.

B. ASIC with on-chip Learning: Loihi

Loihi [105] is a digital neuromorphic chip recently developed by Intel. Loihi is fabricated in Intel's 14-nm process. A Loihi chip contains 128-neuromorphic cores totaling 130,000 artificial current-based (CUBA) leaky-integrate-and-fire neurons and 130 million synapses. It also provides a programmable microcode learning engine for on-chip SNN training. A Loihi chip consists of 3 Lakemont cores, which help with implementing advanced learning rules and managing the neuromorphic cores. The Loihi design supports scaling up to 4,096 on-chip cores and 16,384 chips.

1) Neuron and Synapse Implementation: Loihi adopts a variation of the CUBA LIF model that has two internal state variables, the synaptic response current $u_i(t)$ and the membrane potential $v_i(t)$. The synaptic response current is given by the sum of filtered input spike trains and a constant bias current:

$$u_{i(t)} = \sum_{j \neq i} w_{i,j} (\alpha_u * S_j)(t) + b_i \tag{16}$$

where w_{ij} is the synaptic weight from neuron j to i, $\alpha_u(t) = \tau_u^{-1} \exp(-t/\tau_u) H(t)$ is the synaptic filter impulse response parameterized by the time constant τ_u with $H(t)$ the unit step function, $S_j(t)$ is the input spike train, "*" indicates the convolution operation, and b_i is a constant bias. As shown in Eq. 16, the same kernel $\alpha_u(t)$ is used by all synapses of the postsynaptic neuron u. The synaptic current is further integrated into the membrane potential based on Eq. 17, and the neuron spikes when its membrane potential passes its firing threshold θ_i.

$$v_i(t+1) = -\frac{1}{\tau_v} v_i(t) + u_i(t) - \theta_i S_i(t) \tag{17}$$

where $S_i(t)$ is the output spike of the neuron. As shown in Eq. 17, the integration is leaky, as captured by the time constant τ_v. v_i is initialized with a value less than θ_i, and is reset by θ_i after a spiking event occurs.

Each synapse in Loihi is configured by a 5-tuple: $(i, j, weight, delay, tag)$, where i, j are the source and destination neuron indices of the synapse, and $weight$, $delay$ and tag are integer-valued properties of the synapse. Synaptic delays enable advanced temporal codes by delaying the accumulation of an incoming spike, while tags are useful as an additional scratch variable within the learning engine. Each synapse also associates with multiple presynaptic traces, and whereas compartment with postsynaptic traces. They use different exponential smoothing parameters with decay α and impulse magnitude δ and are evaluated as follows:

$$x[t] = \alpha \cdot x[t-1] + \delta \cdot S[t] \tag{18}$$

where $x[t]$ is the trace variable and $S[t]$ is the incoming spike train. The traces are used by the learning engine as input variables for synaptic adaptation. Loihi supports in-hardware adaptation for all three synaptic variables, weight, delay and tag. The locality constraint is satisfied during the procedure. The weight (delay, tag) can only be accessed and modified by the postsynaptic neuron, based only on locally available information, such as the spike trains from the presynaptic (source) and postsynaptic (destination) neurons. The functional form of adaptation rules is described in sum-of-products form in terms of microcode operations associated with the synapse:

$$z = z + \sum_{i=1}^{N_p} A_i \prod_{j=1}^{n_i} (x_{i,j} + C_{i,j}) \tag{19}$$

where z is the transformed synaptic variable $(weight, delay$ or $tag)$, $x_{i,j}$ refers to some selected input traces available to the learning engine, and $C_{i,j}$ and A_i are microcode-specified signed constants [105]. Based on Eq. 19, the learning engine supports simple pairwise STDP rules and also much more complicated rules such as triplet STDP [265], [266], reinforcement learning with synaptic tag assignments [267], and complex rules that reference both rate averaged and spike-timing traces.

2) Architecture and Implementation: The Loihi processor is a digital and functionally deterministic neuromorphic chip. It was implemented in an asynchronous bundled data design style allowing for event-driven communication through spikes with maximal activity gating during idle periods. It was fabricated in Intel's 14nm FinFET process. The chip has a die area of 60 mm^2 containing 2.07 billion transistors and consists of 128 neuromorphic cores and three x86 cores. Loihi includes a total of 16MB of synaptic memory. It boasts a maximum synaptic density of 2.1 million unique synaptic variables per mm^2 with its densest 1-bit synapse format and maximum neuron density of 2,184 per mm2. After adjusting for the benefit from the advanced technology, this comes to a 2$\times$ reduction in the neuron density in comparison to TrueNorth, which can be interpreted as the cost of Loihi's greatly expanded feature set.

Each neuromorphic core in Loihi implements 1,024 primitive spiking neural units called compartments, which can be grouped into sets of trees constituting neurons. The architecture memory for the storage of configuration and state variables for the compartments and the associated connectivity (fan-in and fan-out) are shared in a core. Every algorithmic timestep, the state variables are updated in a time-multiplexed, pipelined manner. When a neuron's activation exceeds the threshold, it generates a spike message that is routed to the fan-out compartment in one or multiple destination cores.

An asynchronous network-on-chip (NoC) forms the backbone for communication between the cores. All communication between cores occurs in the form of packetized messages. The different types of communication messages include core management and x86-to-x86 messaging, spike messages, and barrier messages for time synchronization between cores. The NoC distributes the communication messages according to the dimension-order routing. The NoC itself only supports unicast distributions. To multicast spikes, the output process of each core iterates over a list of destination cores for a firing neuron's fan-out connections and sends one spike per core.

The host CPU, the on-chip x86 processors and the neural cores can communicate with each other using any type of messages. For off-chip communication over a second-level network, messages may be hierarchically encapsulated. The mesh protocol allows for scaling to 4096 on-chip cores and up to 16,384 chips.

3) Supporting Software/Software Ecosystem: Loihi provides a Python-based API that can be used to specify complex SNN topologies and to program custom learning rules. It also provides a compiler and runtime library for building and executing SNNs on Loihi. The API utilizes core primitives: neuronal compartments and synaptic connections as means of defining SNN topology, synaptic traces and a neuron model to describe SNN dynamics, and synaptic learning rules. Thus, enabling the programmers to implement SNNs in an intuitive way without requiring intimate knowledge of its architectural details. The compiler takes an SNN implementation and produces a binary byte stream in three steps: preprocessing, resource allocation, and code generation. Due to the support of more complex neuron and synapse models, the application of Loihi is more diversified. It has been applied to accelerate the process of Locally Competitive Algorithm for LASSO [105], Neural Engineering Framework (NEF) [52], Stochastic SNNs for solving Constraint Satisfaction Problems [268], Parallel graph search [269] and Random diffusion walkers [270]. It has also been used to implement biological inspired systems such as Olfaction-inspired rapid learning [271]. Dynamic Neural Fields [272], SLAM [72], Evolutionary search [273] are fields to which Loihi is being applied. It has also been used to implement deep SNN for conventional machine learning applications such as classification or prediction. For these applications, Nengo is used to convert a DNN to SNN [274], [275].

C. Analog/Mixed-signal System: BrainScaleS

Analog/mixed-signal design has always played an important role in neuromorphic computing due to its analogue to biological systems. After modeling the neurons and synapses using circuits consisting of resistors, and capacitors/inductors, basic operations of neural computation such as conservation of charge, amplification, exponentiation, integration, thresholding, etc., can be naturally emulated [4]. Such analog/mixed-signal design has the potential to achieve higher speedup and energy efficiency than digital systems. Some representative analog/mixed-signal neuromorphic computing systems are: BrainScaleS, BrainDrop, NeuroGrid, DYNAP-SEL etc. Though these systems differ in various aspects, they share the same design philosophy. They all use analog circuits to implement neuron and synapse models for efficient computation and implement control, on-chip communication, I/O, data storage using digital circuits. They also adopt multi-core architecture and NoC for parallelism and scalability. In this section, we take BrainScaleS as an example to introduce neuromorphic computing hardware using analog/mixed-signal design.

BrainScaleS is a part of Human Brain Project's (HBP) neuromorphic computing platform. HBP is a brain research initiative supported by the European Union, aimed at facilitating research of human brain related areas, such as neuroscience, medical research, cognitive science as well as brain-inspired computing technologies [276], [277]. HBP neuromorphic computing platform offers two complementary systems: BrainScaleS and SpiNNaker. BrainScaleS implements neuron and synapse models using analog circuits, enabling low power and high speed at the cost of flexibility. SpiNNaker, which will be discussed in Section IV-D, on the other hand, is based on general purpose ARM processors, providing flexible functionalities.

1) SNN models: BrainScales implements an exponential integrate and fire model (AdExp) [278], [279] as below:

$$
\begin{aligned}
-C_m \frac{\mathrm{d}V}{\mathrm{d}t} &= g_l(V - E_l) - g_l \delta_{th} \exp\left(\frac{V - V_{th}}{\delta_{th}}\right) \\
&\quad + g_e(t)(V - E_e) + g_i(t)(V - E_i) + wt
\end{aligned}
\tag{20}
$$

$$
-\tau_w \frac{\mathrm{d}w}{\mathrm{d}t} = w - a(V - E_l) \tag{21}
$$

$$
w \leftarrow w + b \quad \text{upon generating a spike} \tag{22}
$$

Where C_m, g_l, E_l, E_e and E_i are the membrane capacity, leakage conductance, leakage, excitatory and inhibitory reversal potentials respectively [280]. $g_e(t)$ and $g_i(t)$ represent the total excitatory and inhibitory synaptic conductance. V_{th} is the threshold, when $V > V_{th}$, the AdExp neuron potential can develop to infinity rapidly, δ_{th} determines sharpness of the procedure. Equation 21 depicts the evolution of adaption current. w is increased by b, which is called spike triggered adaption, upon generating a spike. τ_w is a time constant and a is subthreshold adaptation efficacy. By ignoring the exponential term and the adaption, the AdExp models can be simplified to the common leaky integrate and fire model. More details about how the neuron model is implemented can be found in [279].

2) Hardware Platform: The full BrainScaleS-1 system (NM-PM-1) consists of 20 neuromorphic wafer modules and peripheral devices such as support infrastructure for power, communication and analog readout. An additional computer cluster is used to control the wafer modules [281].

The underlying building block of the BrainScaleS system is the High Input Count Analog Neural Network chip (HICANN), which is an uncut 20 cm wafer scale chip fabricated by 180 nm CMOS technology [280]. HICANN adopts mixed signal design. Computations of neurons and synapses are carried out by analog circuits; weight storage, control and communication are implemented by digital circuits. By emulating the neuron and synapse differential equations with analog circuits, power consumption can be reduced by several orders of magnitude, compared with solving the differential equations numerically using digital processors [282].

3) System Architecture : In order to address the high communication throughput required by massive simulation and high acceleration factor, HICANN adopts a unique technique, namely wafer-scale integration. The wafer is not cut into individual chips, but all the chips on the wafer are directly interconnected to provide high connection density [283]. A wafer consists of 56 reticles, each of which consist of 8 analog network chips (ANC) [283]. The major component of ANC is Analog Neural Network Core (ANNCORE), which contains 128k synapses and 512 membrane circuits/ dendrite membrane (DenMem) circuits [283]. Each DenMem circuit is connected to 224 synapses. Multiple DenMem circuits can be grouped together to build a neuron, such that neurons can have a variable number of synapses [280]. Up to 64 DemMem circuits can be grouped together, resulting in a single neuron with 14336 synapses. Each synapse has a 4-bit weight stored in SRAM. Synapse current is generated by DAC.

The fault tolerant nature of biological neural network is preserved by HICANN, and hierarchical programmable topology enables the replacement of individual defect neurons or an entire neuron core.

The communication in BrainScales has a hierarchical architecture. The Layer-1 communication is carried out by a continuous-time serial bus system that enables inter wafer communication between ANCs across the entire wafer. The 521 wires of the Layer-1 bus form 256 differential lanes connected directly to the ANCs. Since the signal has to travel along horizon and vertical buses across the wafer, repeaters are required for signal and timing restoration. Repeaters are placed at the boundaries of each chip. Each repeater consists of a receiver, timing restoration circuit and driver [283]. The packet-based inter-wafer or wafer-to-host communication (i.e. Layer-2) is implemented by dynamic routing chips connected to the wafer surface [178].

4) Supporting Software/Software Ecosystem: BrainScaleS supports PyNN as programming interface. A user can specify the parameters of neurons and define connections and network topology by Python [284], [285]. The existing packages in the PyNN ecosystem can also be used with BrainScales [286].

D. Neuromorphic Super Computing Platform: SpiNNaker

The spiking neural network architecture (SpiNNaker) project is a massively parallel computer system based on general purpose ARM processor, aimed at providing high performance and flexible simulators for neuroscience experiments. Its goal is to simulate up to a billion neurons in real time [122].

1) Hardware Platform: The basic building block of the system is the SpiNNaker chip. A SpiNNaker chip is a custom designed multiprocessor system-on-chip, consisting 18 identical ARM968E-S 32-bit processors clocked at 200 MHz [128]. Each core has 32-kB instruction memory and 64-kB data memory. An off-die 128 MB SDRAM is stacked on the chip [118]. The chip adopts a Globally Asynchronous Locally Synchronous (GALS) architecture. Each core resides in its own clock domain [287].

SpiNNaker chips are mounted on a printed circuit board (PCB), forming a 48-node hexagonal array. A full system can have up to 1200 such boards, resulting in 57K nodes, 1M ARM cores and 7 T bytes of RAM in the entire system [122], [128].

SpiNNaker consists of two different types of networks at different hierarchies. The first one at the lower level is the system NoC, which handles communication inside a chip. The system NoC uses AMBA5 AXI interfaces [287]. It connects the ARM cores and several slave devices, such as system controller, Ethernet media-independent interface controller, off-chip SDRAM etc. [128].

The second is the communication NoC, which is a packet switching fabric responsible for system-wide communications. It transmits packets from one processor to any other processor, which doesn't have to be in the same chip. The Router has six full-duplex links connecting to adjacent chips of directions (North, Northeast, East, South, Southwest, West) to form a 2-D triangular toroidal mesh. In addition, the system configuration and information are also transmitted by communication NoC [128], [287].

2) Neuron Models: SpiNNaker is based on a general purpose CPU, it has higher flexibility than BrainScaleS. The project provides a C-based event-driven programming model: SpiNNaker Application Run-Time Kernel (ARK) and Application Programming Interface (API). The programming model enables modelling of arbitrary neuron and synapse dynamics [132].

Users can write C functions (also called "callbacks") to define a particular task, and then register the function to scheduler specific events, so that the function can be triggered by the event. The events can be arrival of a packet, the completion of a DMA transfer, timer etc. [118], [132]. For example, [132] implemented the Izhikevich neuron model and three different synapse models, i.e. current-based instantaneous spike response model, current- and conductance-based models with first-order response. [288] implemented a leaky integrate and fire model. [289] Implemented stochastic neuron models on SpiNNaker, and [290] provided implementation of current-based leaky integrate and fire neurons and Izhikevich neurons.

3) Supporting Software/Software Ecosystem: SpiNNaker has a relatively well-developed software ecosystem compared with other neuromorphic systems. In addition to the basic SpiNNaker API, [248] introduced the PArtitioning and Configuration MANager (PACMAN), which is an intermediate translation layer that decouples the model from SpiNNaker hardware, such that arbitrary neuron and synapse dynamics, and arbitrary network topologies can be implemented on the SpiNNaker system. Various frontend programming libraries are built upon PACMAN to support SpiNNaker including PyNN [291], Nengo [292], NEST [293], Brian [294], [295], sPyNNaker [290] etc.

E. ANN-SNN Hybrid Design: Tianjic

Tianjic [296] is a 28 nm reconfigurable chip designed by Tsinghua University. It provides a hybrid and synergistic platform for both the Spiking Neural Network model and the Artificial Neural Network Model. The Tianjic chip contains around 40,000 neurons and 10 million synapses. Tianjic's flexible reconfiguration enable this chip to implement most neural networks (fully connected, convolutional, pooling, spiking, etc.) from the same basic topological layer.

1) SNN models: Tianjic supports various neural network algorithms, for the neuromorphic approach, it adopts the Spiking Neural Network (SNN) model. The axon block in the FCore is to memorize the historical spikes or ANN inputs and feed them through connected synapses according to its configuration mode. After receiving signals from synapses, the dendrites block performs either integration (SNN mode) or MAC (multiplication and accumulation) operation (ANN mode). The shared dendrites then deliver the results to soma block. As shown in Equation 23, in SNN mode, the Tianjic chip adopts the leaky Integrate-and-fire (LIF) model, where $V(t)$ denotes the membrane potential in the soma unit. The soma part receives voltage V_Σ coming from dendrites, here V_{r1} is the reset voltage and τ is the time constant.

$$\tau \frac{\mathrm{d}V_i(t)}{\mathrm{d}t} = -\left[V_i(t) - V_{r1}\right] + V_\Sigma \tag{23}$$

2) Hardware Platform: The Tianjic chip is fabricated with 28 nm high performance low power (HLP) process, and it occupies 3.8 × 3.8 mm^2 die area. One Tianjic chip consists of 156 Fcores. For each Fcore, Tianjic chip supports 32 weight indices and 256 fan-ins/fan-outs (N), and the static random-access memory (SRAM) of each Fcores is around 22 KB. Unlike Lohi and TrueNorth, the Tianjic chip adopts synchronous circuits and its clock frequency is 300 MHz. The average power consumption for control, audio and base applications is

400 mW under 0.9 V working voltage. Generally, the Tianjic chip requires 5,050 clock periods to complete a round of computations and communications.

3) Architecture: As discussed in the previous section, Tianjic embraces a 2D mesh many-core architecture to achieve massive parallelism. At the coarse-grained level, developers are able to assign some Fcores to ANN mode and other Fcores to SNN mode concurrently. While at the Fcore block (fine-grained) level, the independently reconfigurable axon and soma enable Tianjic to implement neuromorphic and artificial neural networks. Tianjic chip also supports transition mode between ANN and SNN, that is, when axon and soma are set to different modes, FCore can process the ANN's input in axon block to SNN's output in soma block or receive the SNN's inputs from the axon block and convert them to the ANN's outputs in the soma block. This unique transition mode is hybrid mode.

There are two chunks of Axon memory. When the Axon is assigned to ANN mode, the two chunks are served as a ping pong buffer for ANN's input. In SNN mode, these two chunks are merged to store the temporal spike patterns in a time window. As for the dendrite block, the processing neurons are divided into groups, each group has 24-bit accumulators to support the vector-matrix multiplication (VMM) that can be used in both ANN and SNN modes. The dataflow in the soma block is different in ANN mode and SNN mode: In ANN mode, data flows in 'bias, activation function, output transmission' fashion, and the biased activation value is 25-bit; The dataflow changes to 'potential leakage, spike generation, output transmission' fashion in SNN mode, where the membrane potential is also 25-bit.

4) Communication: The routing packet format is the same for both SNN and ANN interFcore transmission, which consists of control, address, and data segments. The post synaptic axon parses received ANN or SNN signals from soma and renders them to the routing blocks. In ANN mode, the data segment transmits as 8-bit activation while in SNN mode it transmits as nothing (itself is a spike or none). The 1KB routing LUT consisting of address and control segments will route the packet to one of the 5 communication channels: local, eastern, western, southern, and northern.

Tianjic chip adopts conventional P2P [114]routing scheme and adjacent multicast (AMC) routing scheme. The reconfigurable routing table allows each Fcore to connect with any other neuron.

5) Supporting Software/Software Ecosystem: Tianjic's software tool chain supports the deployment of various SNN and ANN models. To reduce the latency of the application, Tianjic developed several software techniques, including but not limited to unified abstraction for programming and an automatic compiler for mapping hardware. The software tool chain also supports direct training and indirect training for neural networks. The direct training deploys a spatiotemporal back-propagation algorithm to train the network on chip. The indirect training uses a trained ANN and converts it to SNN.

6) Applications: Tianjic has been tested for several computer vision tasks, such as MNIST detection. To demonstrate that one Tianjic chip can handle complex biological plausible neural networks in parallel, The Tianjic team designed an unmanned bicycle experiment. The experiment requires the chip to handle obstacle avoidance, real-time object detection, voice recognition and decision-making with different neural networks. For example, SNN is utilized for voice recognition, CNN is used for object detection and CANN [297] is used for target tracking.

In addition to the aforementioned systems, many other large-scale neuromorphic computing platforms have been playing an important role in machine intelligence and computational neuroscience. Table III provides a more comprehensive comparison of the technology and performance of the large-scale neuromorphic systems that are currently active.

V. Outlook

The function and behavior of biological neural systems inspire the third generation of neural networks, i.e. Spiking Neural Networks (SNN). While moving from biological system to software simulation deprives the energy efficiency that the brain promises due to the inherent limitations of the Von Neumann architecture in general purpose computers, this challenge has been tackled by the emerging neuromorphic hardware. In this work, we discussed the various aspects of neuromorphic systems, such as the computing models and their design considerations, as well as hardware platform and communication systems. We have also discussed various neuromorphic systems, which have provided not only solid foundations for SNN hardware implementation, but also exciting computing platforms for a variety of research fields helping to push forward the frontier of computational neuroscience and machine intelligence. However, there are still many areas waiting to be explored. The research needs in neuromorphic computing can be categorized into three areas 1) algorithm and computational model. 2) hardware architecture. 3) emerging device technologies.

A. Challenges in Model and Algorithm Design

On the algorithmic level, in-hardware learning is still a major road block.

The capability of incrementally augmenting its knowledge base during run time and adapting itself to the changing environment is crucial to an intelligent system. In digital neuromorphic hardware, the memory capacity not only limits the network size, but also the size of the neuron/synapse state variables and the data precision. This restricts the complexity of learning rules that can be implemented on the hardware. While advances have been made in approximating the backpropagation algorithm in SNNs in recent years [111], the quality of learning suffers from low precision of data and weight representations. How to improve the accuracy of supervised learning under limited precision of weight and neuron activities is one of the problems that need to be solved with high urgency.

Alongside further investigating the traditional backpropagation algorithms, application developers should look beyond them so as to potentially revolutionize online learning. It is widely accepted that the biological learning rules, such

TABLE II: Design choices in the large-scale neuromorphic systems.

	Neuron	**Synapse**	**Implementation Choice**	**Architecture**	**Software Support**
TrueNorth	Classic LIF	Binary with a choice of four 8-bit weights	Digital	256x256 crossbar per core, 64x64 core array	Matlab-based object-oriented Corelet language
Loihi	CUBA LIF	Variable precision weight, allows PSP with exponential kernel filter	Digital	No crossbar, 1048 neurons per core, 128 cores	Python-based API NxSDK, also supported by Nengo
SpiNNaker	Any	Any	Digital, Multiprocessor SOC	-	SpiNNaker API, PyNN, Nengo, NEST, Brian, sPyNNaker
BrainScaleS	Exponential IF (AdExp)	-	Analog/ Mixed signal	A wafer with 56x8 ANC containing ANNCORE, each of which has 128k synapse and 512 membrane circuits	PyNN
Tianjic	Classic LIF	-	Digital	2D mesh many-core with 156 Fcores, each with 32 wieght index and 256 fan-ins/fan-outs	-

TABLE III: Comparison of the large-scale neuromorphic systems.

Neuromorphic Chip	**TrueNorth**	**SpiNNaker**	**Loihi**	**BrainScaleS**	**Neurogrid**	**Braindrop**	**Dynap-SEL**	**Tianjic**
Implementation	Digital	Digital	Digital	Analog	Analog	Analog	Mixed-signal	Digital
Technology	28 nm CMOS	ARM968 130 nm CMOS	14 nm CMOS	180 nm CMOS	180 nm CMOS	28 nm CMOS	180 nm CMOS	28 nm CMOS
# transistors	5.4 B	100 M	2.07 B	15 M	23 M			
Neurons per Core	256	$\sim$1k	max 1024	8 to 512	65k	4096	1024	16
Synapses per Core	256x256	$\sim$1M	$\sim$16k	$\sim$130k	100M	64k	64k	22k
Cores per Chip	4,096	16	128	352	16	16	4	156
Chip Area (mm2)	430	102	60	50	168		43.79	14.44
Energy/SOP (pJ)	26	10000	23.6	100	100	0.38	17	0.95
NoC	2D mesh unicast	2D mesh multicast	2D mesh unicast	Hierarchical	Tree multicast	Tree Multicast	Hierarchical 2D mesh multicast	Hierarchical 2D mesh multicast
Packet Size (bits)	32	40 + optional 32		30	12		20	
Time	Discretized	Discretized	Discretized	Discretized	Real time	Real time	Real time	Real time
Neuron Update	Time Multiplexed	Time Multiplexed	Time Multiplexed	Real time	Real time	Real time	Real time	Real time
Bio-Plausibility	Low	Medium	Medium	High	High	High	High	Low
Simulation Time	1x to 21x real-time	Real-time	>Real-time but variable	104x to 105x	Real-time	70 MHz clock	Real-time	300 MHz
On-Chip Learning	No	Yes	Yes	Yes	No	Yes	Yes	No

as STDP, is unsupervised and local. How to achieve useful machine intelligence using the unsupervised local learning is another area to be explored. This may require novel network architectures that provide local feedback or reward signals during the learning process. Since unsupervised learning in general leads to associative memories, a study on the application development and learning capacity of associative memory is worthwhile.

Finally, like all online learning algorithms, the online learning of SNN will also suffer from catastrophic forgetting and slow convergence. The low data precision in SNN deprives us the flexibility of controlling the weights precisely. Hence, techniques such as meta learning, which carefully move the synaptic weights to a specific combination that works for multiple input domains, may not be applicable for SNN. New techniques to improve the quality of online learning must be studied.

B. Challenges in Architectural Design

At the architecture level, the challenges come from off-chip memory access latency, on-chip memory capacity, highly diverse SNN models, reconfigurability, massive connection, neuron density and network parallelism. The architectural design has to balance these divergent and tightly coupled aspects. The higher degree of flexibility and reconfigurability comes at the cost of additional hardware cost. For example, Loihi suffers a $2\times$ reduction in the neuron density compared with TrueNorth after process normalization [105]. SpiNNaker achieves even higher flexibility as it adopts general purpose ARM core and off-chip storage. To mitigate the memory access latency, SpiNNaker stacks the SDRAM on the chip. Digital designs such as Loihi, TrueNorth, and SpiNNaker all work at the speed comparable to a biological system, while BrainScaleS adopts an analogue design, and it achieves $10,000\times$ speedup over biological speed [298]. These design trade-offs are made to serve specific purposes. Loihi and TrueNorth are mainly designed for machine learning applications, hence use relatively simple models. SpiNNaker is designed as a super computing system for various biological research, hence it uses an ARM core rather than a model-specific core to guarantee flexibility.

Although cost, flexibility, performance, and energy dissipation are always contradictory goals during the hardware design, a better architecture can push the design point for a more efficient trade-off. Optimized resource allocation and scheduling that maps neurons to physical cores while maintaining workload balance and minimum communication will further help to improve the performance and lower the energy dissipation.

C. Emerging Devices

At the device level, the emerging technologies in nano devices and materials provide a potential for extremely small, ultra-fast and extremely low-lower neuromorphic hardware if they are successfully married with suitable algorithms. These works share similar ideas as analog/mixed-signal design, i.e., using the physical process to naturally and efficiently emulate neuron and synapse dynamics.

In his work [299], Chua hypothesized the existence of the missing element, memristor, defined by relation between flux-linkage ϕ and charge q [300]:

$$\mathrm{d}\phi = M(q)\mathrm{d}q \tag{24}$$

where $M(q)$ is a function of the amount of charge q flowed through it. Such a memristor behaves like a non-linear resistor with memory [299].

The synaptic weights in biological systems can be adjusted by the ionic flow. This is analogous to the resistance of the memristor, which can be adjusted by the charge or flux, hence [301] demonstrated that the synapse function can be implemented by a memristor. Furthermore, it showed that STDP can be achieved by a hybrid system, which consists of CMOS neurons and memristive synapses. Since then, the memristor has attracted attentions as a promising implementation technique for neuromorphic systems [302]–[307] [217], [308]–[311]. Most of these works adopt a memristor crossbar as it is capable of providing a high density connection and efficient implementation of matrix-vector multiplication [303], [312], [313] and can be used as an accelerator for neuromorphic computing [310], [314]–[316]. How to realize synaptic plasticity has also drawn a lot of interest [307], [317]–[323]. [318] built a single layer perceptron and implemented in situ training by the delta rule. [324] realized triplet STDP learning rule on memristors. [306] also demonstrated the feasibility of implementing ReLU neuron, convolution layer, fully connected layer and unsupervised synaptic weight update on memristor arrays.

A Photonic device is another promising direction for their ultra-fast operation speed and virtually unlimited bandwidth [325]–[336]. Recent works show that it is feasible to implement synapses and neurons by photonic devices. [325] implemented synapses in the optical domain via a photonic integrated-circuit based on phase-change materials (PCMs) cells. Because the PCM can be adjusted by optical pulses, the PCM cell serves as non-volatile photonic memories. Synaptic plasticity is also demonstrated [325]. [326] realized a scalable all-optical spiking neural network circuit. A network of four input neurons, three hidden-layer neurons and two output neurons were built upon the proposed circuit. The network demonstrated capability of pattern recognition in the optical domain. [330] implemented a neuromorphic photonic network to solve an ordinary differential equation system called a Lorenz attractor, and it achieved 294 $\times$ speedup compared to a CPU baseline. Though photonic neuromorphic computing is still far from practical, it has the potential to exceed electronic devices' performance by many orders of magnitude [337].

While the neuromorphic systems implemented using aforementioned emerging device technologies have demonstrated great potentials, they also face significant challenges. How to improve their scalability, flexibility and reliability will continue to be the research direction in the future.

VI. Conclusions

As a bio-inspired computing paradigm, neuromorphic computing has great potentials in accelerating computational neuroscience, and enabling energy efficient solutions for machine intelligence. Due to its unique way of encoding and processing information, it is also believed to be particularly promising for sensor and control-based applications that interact with the physical environment. In this survey, we reviewed different computation models, learning algorithms, information coding schemes, and hardware architectures of neuromorphic computing. With more and more research efforts in academia and industry, we anticipate that breakthroughs in more reliable learning algorithms and more efficient implementations will be seen in the near future.

Acknowledgment

This work is partially supported by the National Science Foundation I/UCRC ASIC (Alternative Sustainable and Intelligent Computing) Center (CNS-1822165).

The work was received and approved for public release by AFRL on August 24th, 2021, case number AFRL-2021-2820. Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

References

[1] C. A. Mead and M. A. Mahowald, "A silicon model of early visual processing," *Neural networks*, vol. 1, pp. 91–97, 1988.

[2] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, pp. 1629–1636, 1990.

[3] C. Mead and M. Ismail, *Analog VLSI implementation of neural systems*. Springer Science & Business Media, 2012, vol. 80.

[4] R. Douglas, M. Mahowald, and C. Mead, "Neuromorphic analogue VLSI," *Annual review of neuroscience*, vol. 18, pp. 255–281, 1995.

[5] C. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[6] P. E. Hasler, C. Diorio, B. A. Minch, and C. Mead, "Single transistor learning synapses," in *Advances in neural information processing systems*. Advances in neural information processing systems, 1995, pp. 817–824.

[7] M. A. C. Maher, S. P. Deweerth, M. A. Mahowald, and C. A. Mead, "Implementing neural architectures using analog VLSI circuits," *IEEE transactions on circuits and systems*, vol. 36, pp. 643–652, 1989.

[8] R. F. Lyon and C. Mead, "An analog electronic cochlea," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1119–1134, 1988.

[9] A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasovic, R. E. Jenkins, and K. Strohbehn, "Current-mode subthreshold MOS circuits for analog VLSI neural systems," *IEEE Transactions on neural networks*, vol. 2, pp. 205–213, 1991.

[10] C. Diorio, P. Hasler, A. Minch, and C. A. Mead, "A single-transistor silicon synapse," *IEEE transactions on Electron Devices*, vol. 43, pp. 1972–1980, 1996.

[11] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, pp. 1659–1671, 1997.

[12] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[13] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[14] E. Hunsberger and C. Eliasmith, "Spiking deep networks with lif neurons," *arXiv preprint arXiv:1510.08829*, 2015.

[15] Q. Xu, Y. Qi, H. Yu, J. Shen, H. Tang, and G. Pan, "CSNN: An Augmented Spiking based Framework with Perceptron-Inception." in *IJCAI*. IJCAI, 2018, pp. 1646–1652.

[16] W. Wang, S. Hao, Y. Wei, S. Xiao, J. Feng, and N. Sebe, "Temporal Spiking Recurrent Neural Network for Action Recognition," *IEEE Access*, vol. 7, pp. 117 165–117 175, 2019.

[17] L. Shi, J. Pei, N. Deng, D. Wang, L. Deng, Y. Wang, Y. Zhang, F. Chen, M. Zhao, S. Song *et al.*, "Development of a neuromorphic computing system," in *2015 IEEE International Electron Devices Meeting (IEDM)*. 2015 IEEE International Electron Devices Meeting (IEDM), 2015, pp. 4–3.

[18] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[19] S. M. Schuetze, "The discovery of the action potential," *Trends in Neurosciences*, vol. 6, pp. 164–168, 1983.

[20] E. R. Kandel, J. H. Schwartz, T. M. Jessell, D. of Biochemistry, M. B. T. Jessell, S. Siegelbaum, and A. J. Hudspeth, *Principles of neural science*. McGraw-hill New York, 2000, vol. 4.

[21] C. S. Sherrington, "The central nervous system," *A text book of Physiology*, vol. 929, 1897.

[22] M. R. Bennett, "The early history of the synapse: from plato to sherrington," *Brain research bulletin*, vol. 50, pp. 95–118, 1999.

[23] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, pp. 500–544, 1952.

[24] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysical journal*, vol. 1, p. 445, 1961.

[25] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophysical journal*, vol. 35, pp. 193–213, 1981.

[26] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, pp. 1569–1572, 2003.

[27] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input," *Biological cybernetics*, vol. 95, pp. 1–19, 2006.

[28] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[29] H. Fang, Z. Mei, A. Shrestha, Z. Zhao, Y. Li, and Q. Qiu, "Encoding, model, and architecture: Systematic optimization for spiking neural network in FPGAs," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[30] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris *et al.*, "Simulation of networks of spiking neurons: a review of tools and strategies," *Journal of computational neuroscience*, vol. 23, no. 3, pp. 349–398, 2007.

[31] J. Köhn and F. Wörgötter, "Employing the z-transform to optimize the calculation of the synaptic conductance of NMDA and other synaptic channels in network simulations," *Neural Computation*, vol. 10, no. 7, pp. 1639–1651, 1998.

[32] S. Johnston, G. Prasad, L. Maguire, and M. McGinnity, "Comparative investigation into classical and spiking neuron implementations on FPGAs," in *International Conference on Artificial Neural Networks*. Springer, 2005, pp. 269–274.

[33] H. Fang, A. Shrestha, Z. Zhao, and Q. Qiu, "Exploiting Neuron and Synapse Filter Dynamics in Spatial Temporal Learning of Deep Spiking Neural Network," *arXiv preprint arXiv:2003.02944*, 2020.

[34] H. Fang, A. Shrestha, Z. Zhao, Y. Li, and Q. Qiu, "An event-driven neuromorphic system with biologically plausible temporal dynamics," in *38th IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2019*. 38th IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2019, 2019, p. 8942083.

[35] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, pp. 54–66, 2015.

[36] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing. 2016," *Preprint on ArXiv. http://arxiv. org/abs/1603.08270. Accessed*, vol. 27, 2016.

[37] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in neural information processing systems*. Advances in neural information processing systems, 2015, pp. 1117–1125.

[38] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, and F. Morgan, "Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers," *Neural networks*, vol. 33, pp. 42–57, 2012.

[39] G. Liu, P. Camilleri, S. Furber, and J. Garside, "Network traffic exploration on a many-core computing platform: SpiNNaker real-time traffic visualiser," in *2015 11th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, 2015, pp. 228–231.

[40] J. Navaridas, L. A. Plana, J. Miguel-Alonso, M. Luján, and S. B. Furber, "Spinnaker: impact of traffic locality, causality and burstiness on the performance of the interconnection network," in *Proceedings of the 7th ACM international conference on Computing frontiers*. Proceedings of the 7th ACM international conference on Computing frontiers, 2010, pp. 11–20.

[41] P. Reinagel and R. C. Reid, "Temporal coding of visual information in the thalamus," *Journal of Neuroscience*, vol. 20, pp. 5392–5400, 2000.

[42] F. Theunissen and J. P. Miller, "Temporal encoding in nervous systems: a rigorous definition," *Journal of computational neuroscience*, vol. 2, pp. 149–162, 1995.

[43] R. C. Decharms and A. Zador, "Neural representation and the cortical code," *Annual review of neuroscience*, vol. 23, pp. 613–647, 2000.

[44] B. Tripp and C. Eliasmith, "Neural populations can induce reliable postsynaptic currents without observable spike rate changes or precise spike timing," *Cerebral Cortex*, vol. 17, pp. 1830–1840, 2007.

[45] A. Borst and F. E. Theunissen, "Information theory and neural coding," *Nature neuroscience*, vol. 2, pp. 947–957, 1999.

[46] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *science*, vol. 319, pp. 1108–1111, 2008.

[47] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3227–3235, 2017.

[48] S. Park, S. Kim, B. Na, and S. Yoon, "T2FSNN: deep spiking neural networks with time-to-first-spike coding," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[49] L. Zhang, S. Zhou, T. Zhi, Z. Du, and Y. Chen, "Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1319–1326.

[50] Z. Li, B. Yan *et al.*, "Resipe: Reram-based single-spiking processing-in-memory engine," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[51] A. Zador, "Impact of synaptic unreliability on the information transmitted by spiking neurons," *Journal of neurophysiology*, vol. 79, pp. 1219–1229, 1998.

[52] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.

[53] Y. Zheng, S. Li, R. Yan, H. Tang, and K. C. Tan, "Sparse temporal encoding of visual features for robust object recognition by spiking neurons," *IEEE transactions on neural networks and learning systems*, vol. 29, pp. 5823–5833, 2018.

[54] Z. Pan, J. Wu, M. Zhang, H. Li, and Y. Chua, "Neural Population Coding for Effective Temporal Classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–8.

[55] R. Xiao, R. Yan, H. Tang, and K. C. Tan, "A spiking neural network model for sound recognition," in *International Conference on Cognitive Systems and Signal Processing*. International Conference on Cognitive Systems and Signal Processing, 2016, pp. 584–594.

[56] J. Wu, Y. Chua, and H. Li, "A biologically plausible speech recognition framework based on spiking neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.

[57] R. Gütig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing–based decisions," *Nature neuroscience*, vol. 9, pp. 420–428, 2006.

[58] R. Florian, "The chronotron: a neuron that learns to fire temporally-precise spike patterns," *Nature Precedings*, pp. 1–1, 2010.

[59] S. B. Shrestha and G. Orchard, "Slayer: spike layer error reassignment in time," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 1419–1428.

[60] A. Shrestha, K. Ahmed, Y. Wang, D. P. Widemann, A. T. Moody, Van, C. E. Brian, and Q. Qiu, "A spike-based long short-term memory on a neurosynaptic processor," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2017, pp. 631–637.

[61] A. Shrestha, K. Ahmed, Y. Wang, D. P. Widemann, A. T. Moody, B. C. Van Essen, and Q. Qiu, "Modular Spiking Neural Circuits for Mapping Long Short-Term Memory on a Neurosynaptic Processor," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 782–795, 2018.

[62] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Advances in Neural Information Processing Systems*. Advances in Neural Information Processing Systems, 2018, pp. 787–797.

[63] A. Polepalli, N. Soures, and D. Kudithipudi, "Digital neuromorphic design of a liquid state machine for real-time processing," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*. 2016 IEEE International Conference on Rebooting Computing (ICRC), 2016, pp. 1–8.

[64] A. Polepalli, N. Soures, and D. Kudithipudi, "Reconfigurable Digital Design of a Liquid State Machine for Spatio-Temporal Data," in *Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication*, 2016, p. 15.

[65] D. Kudithipudi, Q. Saleh, C. Merkel, J. Thesing, and B. Wysocki, "Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing," *Frontiers in neuroscience*, vol. 9, p. 502, 2016.

[66] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. V. Campenhout, "Compact hardware liquid state machines on FPGA for real-time speech recognition," *Neural networks*, vol. 21, pp. 511–523, 2008.

[67] Q. Wang, Y. Li, and P. Li, "Liquid state machine based pattern recognition on FPGA with firing-activity dependent power gating and approximate computing," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 361–364.

[68] Y. Yi, Y. Liao, B. Wang, X. Fu, F. Shen, H. Hou, and L. Liu, "Fpga based spike-time dependent encoder and reservoir design in neuromorphic computing processors," *Microprocessors and Microsystems*, vol. 46, pp. 175–183, 2016.

[69] A. Zhang, W. Zhu, and J. Li, "Spiking echo state Convolutional Neural Network for Robust Time Series Classification," *IEEE Access*, vol. 7, pp. 4927–4935, 2018.

[70] K. Ahmed, A. Shrestha, Q. Qiu, and Q. Wu, "Probabilistic inference using stochastic spiking neural networks on a neurosynaptic processor," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 4286–4293.

[71] K. Ahmed, A. Shrestha, and Q. Qiu, "Simulation of bayesian learning and inference on distributed stochastic spiking neural networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 1044–1051.

[72] R. Kreiser, A. Renner, Y. Sandamirskaya, and P. Pienroj, "Pose estimation and map formation with spiking neural networks: towards neuromorphic slam," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 2159–2166.

[73] K. Hardcastle, S. Ganguli, and L. M. Giocomo, "Cell types for our sense of location: where we are and where we are going," *Nature neuroscience*, vol. 20, p. 1474, 2017.

[74] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, pp. 59–69, 1982.

[75] G. J. Goodhill, "Contributions of theoretical modeling to the understanding of neural map development," *Neuron*, vol. 56, pp. 301–311, 2007.

[76] K. Ahmed, A. Shrestha, Y. Wang, and Q. Qiu, "System Design for In-Hardware STDP Learning and Spiking Based Probablistic Inference," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 272–277.

[77] A. Shrestha, K. Ahmed, Y. Wang, and Q. Qiu, "Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1999–2006.

[78] J. A. Gottfried, "Central mechanisms of odour object perception," *Nature Reviews Neuroscience*, vol. 11, pp. 628–641, 2010.

[79] D. A. Wilson and R. M. Sullivan, "Cortical processing of odor objects," *Neuron*, vol. 72, pp. 506–519, 2011.

[80] B. A. Kaplan and A. Lansner, "A spiking neural network model of self-organized pattern recognition in the early mammalian olfactory system," *Frontiers in neural circuits*, vol. 8, p. 5, 2014.

[81] H.-Y. Hsieh and K.-T. Tang, "VLSI implementation of a bio-inspired olfactory spiking neural network," *IEEE transactions on neural networks and learning systems*, vol. 23, pp. 1065–1073, 2012.

[82] B.-Z. Li, S. H. Pun, W. Feng, M. I. Vai, A. Klug, and T. C. Lei, "A Spiking Neural Network Model Mimicking the Olfactory Cortex for Handwritten Digit Recognition," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2019, pp. 1167–1170.

[83] B. Grothe, “New roles for synaptic inhibition in sound localization,” *Nature Reviews Neuroscience*, vol. 4, pp. 540–550, 2003.

[84] R. Shi and T. Horiuchi, “A VLSI model of the bat lateral superior olive for azimuthal echolocation,” in *2004 IEEE International Symposium on Circuits and Systems*, vol. 4, 2004, pp. IV–900.

[85] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, “A spiking neural network framework for robust sound classification,” *Frontiers in neuroscience*, vol. 12, p. 836, 2018.

[86] B. Glackin, J. A. Wall, T. M. McGinnity, L. P. Maguire, and L. J. McDaid, “A spiking neural network model of the medial superior olive using spike timing dependent plasticity for sound localization,” *Frontiers in computational neuroscience*, vol. 4, p. 18, 2010.

[87] C. Eliasmith, M. B. Westover, and C. H. Anderson, “A general framework for neurobiological modeling: An application to the vestibular system,” *Neurocomputing*, vol. 44, pp. 1071–1076, 2002.

[88] C. Eliasmith, “A unified approach to building and controlling spiking attractor networks,” *Neural computation*, vol. 17, pp. 1276–1314, 2005.

[89] E. P. Frady and F. T. Sommer, “Robust computation with rhythmic spike patterns,” *Proceedings of the National Academy of Sciences*, vol. 116, pp. 18 050–18 059, 2019.

[90] M. Giulioni, P. Camilleri, M. Mattia, V. Dante, J. Braun, and P. D. Giudice, “Robust working memory in an asynchronously spiking neural network realized with neuromorphic VLSI,” *Frontiers in neuroscience*, vol. 5, p. 149, 2012.

[91] M. Baudry, “Synaptic plasticity and learning and memory: 15 years of progress,” *Neurobiology of learning and memory*, vol. 70, pp. 113–118, 1998.

[92] A. Citri and R. C. Malenka, “Synaptic plasticity: multiple forms, functions, and mechanisms,” *Neuropsychopharmacology*, vol. 33, pp. 18–41, 2008.

[93] P. J. Sjöström, G. G. Turrigiano, and S. B. Nelson, “Neocortical LTD via coincident activation of presynaptic NMDA and cannabinoid receptors,” *Neuron*, vol. 39, pp. 641–654, 2003.

[94] R. A. Nicoll and D. Schmitz, “Synaptic plasticity at hippocampal mossy fibre synapses,” *Nature Reviews Neuroscience*, vol. 6, pp. 863–876, 2005.

[95] G.-q. Bi and M.-m. Poo, “Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type,” *Journal of neuroscience*, vol. 18, pp. 10 464–10 472, 1998.

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

[97] S. Song, K. D. Miller, and L. F. Abbott, “Competitive hebbian learning through spike-timing-dependent synaptic plasticity,” *Nature neuroscience*, vol. 3, pp. 919–926, 2000.

[98] Q. Yu, H. Li, and K. C. Tan, “Spike timing or rate? Neurons learn to make decisions for both through threshold-driven plasticity,” *IEEE transactions on cybernetics*, vol. 49, pp. 2178–2189, 2018.

[99] R. V. Florian, “Tempotron-like learning with resume,” in *International Conference on Artificial Neural Networks*. International Conference on Artificial Neural Networks, 2008, pp. 368–375.

[100] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, “Span: Spike pattern association neuron for learning spatio-temporal spike patterns,” *International journal of neural systems*, vol. 22, p. 1250012, 2012.

[101] R. Echeveste and C. Gros, “Two-trace model for spike-timing-dependent synaptic plasticity,” *Neural computation*, vol. 27, pp. 672–698, 2015.

[102] R. Kempter, W. Gerstner, and J. L. Van Hemmen, “Hebbian learning and spiking neurons,” *Physical Review E*, vol. 59, no. 4, p. 4498, 1999.

[103] J. Rubin, D. D. Lee, and H. Sompolinsky, “Equilibrium properties of temporally asymmetric Hebbian plasticity,” *Physical review letters*, vol. 86, no. 2, p. 364, 2001.

[104] F. Ponulak and A. Kasiński, “Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting,” *Neural computation*, vol. 22, no. 2, pp. 467–510, 2010.

[105] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, “Loihi: A neuromorphic manycore processor with on-chip learning,” *IEEE Micro*, vol. 38, pp. 82–99, 2018.

[106] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, “Event-driven random back-propagation: Enabling neuromorphic deep learning machines,” *Frontiers in neuroscience*, vol. 11, p. 324, 2017.

[107] A. Tavanaei and A. Maida, “BP-STDP: Approximating backpropagation using spike timing dependent plasticity,” *Neurocomputing*, vol. 330, pp. 39–47, 2019.

[108] A. Shrestha, H. Fang, Q. Wu, and Q. Qiu, “Approximating back-propagation for a biologically plausible local learning rule in spiking neural networks,” in *Proceedings of the International Conference on Neuromorphic Systems*. Proceedings of the International Conference on Neuromorphic Systems, 2019, pp. 1–8.

[109] S. M. Bohte, J. N. Kok, and J. A. La Poutré, “Spikeprop: backpropagation for networks of spiking neurons.” in *ESANN*, vol. 48. Bruges, 2000, pp. 419–424.

[110] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, “Spatio-temporal back-propagation for training high-performance spiking neural networks,” *Frontiers in neuroscience*, vol. 12, p. 331, 2018.

[111] A. Shrestha, H. Fang, D. P. Rider, Z. Mei, and Q. Qiu, “In-Hardware Learning of Multilayer Spiking Neural Networks on a Neuromorphic Processor,” in *Design Automation Conference (DAC)*. Design Automation Conference (DAC), 2021.

[112] S. A. Aamir, Y. Stradmann, P. Müller, C. Pehle, A. Hartel, A. Grübl, J. Schemmel, and K. Meier, “An accelerated LIF neuronal network array for a large-scale mixed-signal neuromorphic architecture,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, pp. 4299–4312, 2018.

[113] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” *Proceedings of the IEEE*, vol. 102, pp. 699–716, 2014.

[114] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, pp. 668–673, 2014.

[115] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, “A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses,” *Frontiers in neuroscience*, vol. 9, p. 141, 2015.

[116] S. Yang, J. Wang, B. Deng, C. Liu, H. Li, C. Fietkiewicz, and K. A. Loparo, “Real-time neuromorphic system for large-scale conductance-based spiking neural networks,” *IEEE transactions on cybernetics*, vol. 49, pp. 2490–2503, 2018.

[117] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman *et al.*, “Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores,” in *The 2013 International Joint Conference on Neural Networks (IJCNN)*. The 2013 International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–10.

[118] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, “The spinnaker project,” *Proceedings of the IEEE*, vol. 102, pp. 652–665, 2014.

[119] R. Araújo, N. Waniek, and J. Conradt, “Development of a dynamically extendable spinnaker chip computing module,” in *International Conference on Artificial Neural Networks*. International Conference on Artificial Neural Networks, 2014, pp. 821–828.

[120] P. U. Diehl and M. Cook, “Efficient implementation of stdp rules on spinnaker neuromorphic hardware,” in *2014 International Joint Conference on Neural Networks (IJCNN)*. 2014 International Joint Conference on Neural Networks (IJCNN), 2014, pp. 4288–4295.

[121] S. Furber and A. Brown, “Biologically-inspired massively-parallel architectures-computing beyond a million processors,” in *2009 Ninth International Conference on Application of Concurrency to System Design*. 2009 Ninth International Conference on Application of Concurrency to System Design, 2009, pp. 3–12.

[122] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, “Overview of the spinnaker system architecture,” *IEEE Transactions on Computers*, vol. 62, pp. 2454–2467, 2012.

[123] S. Furber, “To build a brain,” *IEEE spectrum*, vol. 49, pp. 44–49, 2012.

[124] X. Jin, S. B. Furber, and J. V. Woods, “Efficient modelling of spiking neural networks on a scalable chip multiprocessor,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 2812–2819.

[125] J. C. Knight, P. J. Tully, B. A. Kaplan, A. Lansner, and S. B. Furber, “Large-scale simulations of plastic neural networks on neuromorphic hardware,” *Frontiers in neuroanatomy*, vol. 10, p. 37, 2016.

[126] J. C. Knight and S. B. Furber, “Synapse-centric mapping of cortical models to the SpiNNaker neuromorphic architecture,” *Frontiers in neuroscience*, vol. 10, p. 420, 2016.

[127] X. Lagorce, E. Stromatias, F. Galluppi, L. A. Plana, S.-C. Liu, S. B. Furber, and R. B. Benosman, “Breaking the millisecond barrier on

SpiNNaker: implementing asynchronous event-based plastic models with microsecond resolution," *Frontiers in neuroscience*, vol. 9, p. 206, 2015.

[128] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 1943–1953, 2013.

[129] E. Painkras, L. A. Plana, J. Garside, S. Temple, S. Davidson, J. Pepper, D. Clark, C. Patterson, and S. Furber, "Spinnaker: A multi-core system-on-chip for massively-parallel neural net simulation," in *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*. Proceedings of the IEEE 2012 Custom Integrated Circuits Conference, 2012, pp. 1–4.

[130] L. A. Plana, D. Clark, S. Davidson, S. Furber, J. Garside, E. Painkras, J. Pepper, S. Temple, and J. Bainbridge, "SpiNNaker: design and implementation of a GALS multicore system-on-chip," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 7, pp. 1–18, 2011.

[131] A. D. Rast, X. Jin, F. Galluppi, L. A. Plana, C. Patterson, and S. Furber, "Scalable event-driven native parallel processing: the SpiNNaker neuromimetic system," in *Proceedings of the 7th ACM international conference on Computing frontiers*. Proceedings of the 7th ACM international conference on Computing frontiers, 2010, pp. 21–30.

[132] T. Sharp, L. A. Plana, F. Galluppi, and S. Furber, "Event-driven simulation of arbitrary spiking neural networks on Spinnaker," in *International conference on neural information processing*. International conference on neural information processing, 2011, pp. 424–430.

[133] E. Stromatias, D. Neil, F. Galluppi, M. Pfeiffer, S.-C. Liu, and S. Furber, "Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on spinnaker," in *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8.

[134] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 34, pp. 1537–1557, 2015.

[135] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. B. D. Rubin, R. Manohar, and D. S. Modha, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.

[136] A. S. Cassidy, R. Alvarez-Icaza, F. Akopyan, J. Sawada, J. V. Arthur, P. A. Merolla, P. Datta, M. G. Tallada, B. Taba, A. Andreopoulos *et al.*, "Real-time scalable cortical computing at 46 giga-synaptic OPS/watt with˜ 100× speedup in time-to-solution and˜ 100,000× reduction in energy-to-solution," in *SC14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2014, pp. 27–38.

[137] N. Imam, F. Akopyan, J. Arthur, P. Merolla, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using event-driven qdi circuits," in *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems*. 2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems, 2012, pp. 25–32.

[138] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE custom integrated circuits conference (CICC)*. 2011 IEEE custom integrated circuits conference (CICC), 2011, pp. 1–4.

[139] J.-s. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha *et al.*, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*. 2011 IEEE Custom Integrated Circuits Conference (CICC), 2011, pp. 1–4.

[140] A. Banerjee, S. Kar, S. Roy, A. Bhaduri, and A. Basu, "A current-mode spiking neural classifier with lumped dendritic nonlinearity," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2015 IEEE International Symposium on Circuits and Systems (ISCAS), 2015, pp. 714–717.

[141] M. Ambroise, T. Levi, Y. Bornat, and S. Saighi, "Biorealistic spiking neural network on FPGA," in *2013 47th Annual Conference on Information Sciences and Systems (CISS)*. 2013 47th Annual Conference on Information Sciences and Systems (CISS), 2013, p. 1–6.

[142] 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 transactions on neural networks and learning systems*, vol. 25, pp. 1864–1878, 2014.

[143] M. E. Dean and C. Daffron, "A VLSI design for neuromorphic computing," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2016, pp. 87–92.

[144] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, "A 640m pixel/s 3.65 mw sparse event-driven neuromorphic object recognition processor with on-chip learning," in *2015 Symposium on VLSI Circuits (VLSI Circuits)*, 2015, pp. C50–C51.

[145] A. Nere, U. Olcese, D. Balduzzi, and G. Tononi, "A neuromorphic architecture for object recognition and motion anticipation using burst-STDP," *PloS one*, vol. 7, 2012.

[146] A. Nere, A. Hashmi, M. Lipasti, and G. Tononi, "Bridging the semantic gap: Emulating biological neuronal behaviors with simple digital neurons," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), 2013, pp. 472–483.

[147] J.-s. Seo and M. Seok, "Digital CMOS neuromorphic processor design featuring unsupervised online learning," in *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2015, pp. 49–51.

[148] J. Shen, D. Ma, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, and G. Pan, "Darwin: a neuromorphic hardware co-processor based on spiking neural networks," *Science China Information Sciences*, vol. 59, pp. 1–5, 2016.

[149] R. M. Wang, T. J. Hamilton, J. C. Tapson, and A. van Schaik, "A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks," *Frontiers in neuroscience*, vol. 9, p. 180, 2015.

[150] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory," *ACM SIGARCH Computer Architecture News*, vol. 44, pp. 380–392, 2016.

[151] E. O. Neftci, B. U. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs, "Stochastic synapses enable efficient brain-inspired learning machines," *Frontiers in neuroscience*, vol. 10, p. 241, 2016.

[152] E. Neftci, "Stochastic neuromorphic learning machines for weakly labeled data," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, 2016, pp. 670–673.

[153] P. Knag, C. Liu, and Z. Zhang, "A 1.40 mm 2 141mw 898GOPS sparse neuromorphic processor in 40nm CMOS," in *2016 IEEE symposium on VLSI circuits (VLSI-circuits)*. 2016 IEEE symposium on VLSI circuits (VLSI-circuits), 2016, pp. 1–2.

[154] B. U. Pedroni, S. Das, J. V. Arthur, P. A. Merolla, B. L. Jackson, D. S. Modha, K. Kreutz-Delgado, and G. Cauwenberghs, "Mapping generative models onto a network of digital spiking neurons," *IEEE transactions on biomedical circuits and systems*, vol. 10, pp. 837–854, 2016.

[155] S. Das, B. U. Pedroni, P. Merolla, J. Arthur, A. S. Cassidy, B. L. Jackson, D. Modha, G. Cauwenberghs, and K. Kreutz-Delgado, "Gibbs sampling with low-power spiking digital neurons," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2015 IEEE International Symposium on Circuits and Systems (ISCAS), 2015, pp. 2704–2707.

[156] L. Wan, Y. Luo, S. Song, J. Harkin, and J. Liu, "Efficient neuron architecture for FPGA-based spiking neural networks," in *2016 27th Irish Signals and Systems Conference (ISSC)*. 2016 27th Irish Signals and Systems Conference (ISSC), 2016, pp. 1–6.

[157] H. Fang, A. Shrestha, D. Ma, and Q. Qiu, "Scalable NoC-based neuromorphic hardware learning and inference," in *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.

[158] D. Neil and S.-C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Transactions on Very Large Scale Integration VLSI Systems*, vol. 22, pp. 2621–2628, 2014.

[159] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Networks*, vol. 45, pp. 4–26, 2013.

[160] K. Cheung, S. R. Schultz, and W. Luk, "Neuroflow: a general purpose spiking neural network simulation platform using customizable processors," *Frontiers in neuroscience*, vol. 9, p. 516, 2016.

[161] A. Sripad, G. Sanchez, M. Zapata, V. Pirrone, T. Dorta, S. Cambria, A. Marti, K. Krishnamourthy, and J. Madrenas, "Snava—a real-time multi-FPGA multi-model spiking neural network simulation architecture," *Neural Networks*, vol. 97, pp. 28–45, 2018.

[162] S. Yang, Q. Wu, and R. Li, "A case for spiking neural network simulation based on configurable multiple-FPGA systems," *Cognitive neurodynamics*, vol. 5, p. 301, 2011.

[163] M. Heidarpur, A. Ahmadi, M. Ahmadi, and M. R. Azghadi, "CORDIC-SNN: On-FPGA STDP learning with Izhikevich neurons," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, pp. 2651–2661, 2019.

[164] Y. Liu, S. S. Yenamachintala, and P. Li, "Energy-efficient FPGA spiking neural accelerators with supervised and unsupervised spike-timing-dependent-plasticity," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, pp. 1–19, 2019.

[165] A. Yousefzadeh, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco, "Hardware implementation of convolutional STDP for on-line visual feature learning," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1–4.

[166] A. Yousefzadeh, E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "On practical issues for stochastic STDP hardware with 1-bit synaptic weights," *Frontiers in neuroscience*, vol. 12, p. 665, 2018.

[167] C. Bartolozzi, O. Nikolayeva, and G. Indiveri, "Implementing homeostatic plasticity in VLSI networks of spiking neurons," in *2008 15th IEEE International Conference on Electronics, Circuits and Systems*. 2008 15th IEEE International Conference on Electronics, Circuits and Systems, 2008, pp. 682–685.

[168] J. Schreiter, U. Ramacher, A. Heittmann, D. Matolini, and R. Schuffny, "Analog implementation for networks of integrate-and-fire neurons with adaptive local connectivity," in *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*. Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing, 2002, pp. 657–666.

[169] F. L. M. Huayaney, S. Nease, and E. Chicca, "Learning in silicon beyond STDP: a neuromorphic implementation of multi-factor synaptic plasticity with calcium-based dynamics," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, pp. 2189–2199, 2016.

[170] Y. Meng, K. Zhou, J. J. C. Monzon, and C.-S. Poon, "Iono-neuromorphic implementation of spike-timing-dependent synaptic plasticity," in *2011 Annual international conference of the IEEE engineering in medicine and biology society*. 2011 Annual international conference of the IEEE engineering in medicine and biology society, 2011, pp. 7274–7277.

[171] L. Alvado, J. Tomas, S. R.-L. Masson, and V. Douence, "Design of an analogue ASIC using subthreshold CMOS transistors to model biological neurons," in *Proceedings of the IEEE 2001 Custom Integrated Circuits Conference*, 2001, pp. 97–100.

[172] C. Bartolozzi, S. Mitra, and G. Indiveri, "An ultra low power current-mode filter for neuromorphic systems and biomedical signal processing," in *2006 IEEE Biomedical Circuits and Systems Conference*. 2006 IEEE Biomedical Circuits and Systems Conference, 2006, pp. 130–133.

[173] J. Georgiou, E. M. Drakakis, C. Toumazou, and P. Premanoj, "An analogue micropower log-domain silicon circuit for the Hodgkin and Huxley nerve axon," in *ISCAS99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI*, vol. 2, 1999, pp. 286–289.

[174] K. Nakada, T. Asai, T. Hirose, and Y. Amemiya, "Analog CMOS implementation of a neuromorphic oscillator with current-mode low-pass filters," in *2005 IEEE International Symposium on Circuits and Systems*. 2005 IEEE International Symposium on Circuits and Systems, 2005, pp. 1923–1926.

[175] K. I. Papadimitriou, S.-C. Liu, G. Indiveri, and E. M. Drakakis, "Neuromorphic log-domain silicon synapse circuits obey bernoulli dynamics: a unifying tutorial analysis," *Frontiers in neuroscience*, vol. 8, p. 428, 2015.

[176] T. Yu and G. Cauwenberghs, "Analog VLSI biophysical neurons and synapses with programmable membrane channel kinetics," *IEEE Transactions on Biomedical circuits and Systems*, vol. 4, pp. 139–148, 2010.

[177] J. Binas, G. Indiveri, and M. Pfeiffer, "Spiking analog VLSI neuron assemblies as constraint satisfaction problem solvers," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 2094–2097.

[178] C.-H. Chien, S.-C. Liu, and A. Steimer, "A neuromorphic VLSI circuit for spike-based random sampling," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, pp. 135–144, 2015.

[179] J. Fieres, J. Schemmel, and K. Meier, "Realizing biological spiking network models in a configurable wafer-scale hardware system," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 969–976.

[180] D. H. Goldberg, G. Cauwenberghs, and A. G. Andreou, "Analog VLSI spiking neural network with address domain probabilistic synapses," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems*, vol. 3. ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems, 2001, pp. 241–244.

[181] F. Grassia, T. Lévi, J. Tomas, S. Renaud, and S. Saïghi, "A neuromimetic spiking neural network for simulating cortical circuits," in *2011 45th Annual Conference on Information Sciences and Systems*. IEEE, 2011, pp. 1–6.

[182] E. Neftci, J. Binas, U. Rutishauser, E. Chicca, G. Indiveri, and R. J. Douglas, "Synthesizing cognition in neuromorphic electronic systems," *Proceedings of the National Academy of Sciences*, vol. 110, pp. E3468–E3476, 2013.

[183] D. Querlioz and V. Trauchessec, "Stochastic resonance in an analog current-mode neuromorphic circuit," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), 2013, pp. 1596–1599.

[184] S. Renaud, J. Tomas, Y. Bornat, A. Daouzli, and S. Saïghi, "Neuromimetic ICs with analog cores: an alternative for simulating spiking neural networks," in *2007 IEEE international symposium on circuits and systems*. IEEE, 2007, pp. 3355–3358.

[185] A. Utagawa, T. Asai, T. Hirose, and Y. Amemiya, "An inhibitory neural-network circuit exhibiting noise shaping with subthreshold MOS neuron circuits," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, pp. 2108–2115, 2007.

[186] Y. Wang and S.-C. Liu, "Programmable synaptic weights for an a VLSI network of spiking neurons," in *2006 IEEE International Symposium on Circuits and Systems*, 2006, p. 4.

[187] Y. Wang and S.-C. Liu, "Input evoked nonlinearities in silicon dendritic circuits," in *2009 IEEE International Symposium on Circuits and Systems*. 2009 IEEE International Symposium on Circuits and Systems, 2009, pp. 2894–2897.

[188] L. Zhang, Q. Lai, and Y. Chen, "Configurable neural phase shifter with spike-timing-dependent plasticity," *IEEE Electron Device Letters*, vol. 31, pp. 716–718, 2010.

[189] C. Zhao, J. Li, L. Liu, L. S. Koutha, J. Liu, and Y. Yi, "Novel spike based reservoir node design with high performance spike delay loop," in *Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication*. Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication, 2016, pp. 1–5.

[190] A. Ghani, T. M. McGinnity, L. P. Maguire, and J. Harkin, "Area efficient architecture for large scale implementation of biologically plausible spiking neural networks on reconfigurable hardware," in *2006 International Conference on Field Programmable Logic and Applications*. 2006 International Conference on Field Programmable Logic and Applications, 2006, pp. 1–2.

[191] M. Giulioni, X. Lagorce, F. Galluppi, and R. B. Benosman, "Event-based computation of motion flow on a neuromorphic analog neural platform," *Frontiers in neuroscience*, vol. 10, p. 35, 2016.

[192] M. Giulioni, F. Corradi, V. Dante, and P. D. Giudice, "Real time unsupervised learning of visual stimuli in neuromorphic VLSI systems," *Scientific reports*, vol. 5, p. 14730, 2015.

[193] P. Camilleri, M. Giulioni, V. Dante, D. Badoni, G. Indiveri, B. Michaelis, J. Braun, and P. D. Giudice, "A neuromorphic a VLSI network chip with configurable plastic synapses," in *7th International Conference on Hybrid Intelligent Systems (HIS 2007)*. 7th International Conference on Hybrid Intelligent Systems (HIS 2007), 2007, pp. 296–301.

[194] M. Giulioni, P. Camilleri, V. Dante, D. Badoni, G. Indiveri, J. Braun, and P. D. Giudice, "A VLSI network of spiking neurons with plastic fully configurable "stop-learning" synapses," in *2008 15th IEEE International Conference on Electronics, Circuits and Systems*. 2008 15th

IEEE International Conference on Electronics, Circuits and Systems, 2008, pp. 678–681.

[195] M. Giulioni, M. Pannunzi, D. Badoni, V. Dante, and P. D. Giudice, “Classification of correlated patterns with a configurable analog VLSI neural network of spiking neurons and self-regulating plastic synapses,” *Neural computation*, vol. 21, pp. 3106–3129, 2009.

[196] Q. Sun, F. Schwartz, J. Michel, Y. Herve, and R. D. Molin, “Implementation study of an analog spiking neural network for assisting cardiac delay prediction in a cardiac resynchronization therapy device,” *IEEE transactions on neural networks*, vol. 22, pp. 858–869, 2011.

[197] P. Hafliger, “Adaptive WTA with an analog VLSI neuromorphic learning chip,” *IEEE transactions on neural networks*, vol. 18, pp. 551–572, 2007.

[198] H.-Y. Hsieh and K.-T. Tang, “Hardware friendly probabilistic spiking neural network with long-term and short-term plasticity,” *IEEE transactions on neural networks and learning systems*, vol. 24, pp. 2063–2074, 2013.

[199] F. Grassia, L. Buhry, T. Lévi, J. Tomas, A. Destexhe, and S. Saïghi, “Tunable neuromimetic integrated system for emulating cortical neuron models,” *Frontiers in NEUROSCIENCE*, vol. 5, p. 134, 2011.

[200] F. L. M. Huayaney and E. Chicca, “A VLSI implementation of a calcium-based plasticity learning model,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 373–376.

[201] J. Park, M.-W. Kwon, H. Kim, and B.-G. Park, “Neuromorphic system based on CMOS inverters and Si-based synaptic device,” *Journal of nanoscience and nanotechnology*, vol. 16, pp. 4709–4712, 2016.

[202] S. Saighi, J. Tomas, Y. Bornat, B. Belhadj, O. Malot, and S. Renaud, “Real-time multi-board architecture for analog spiking neural networks,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010, pp. 1939–1942.

[203] S. Millner, A. Hartel, J. Schemmel, and K. Meier, “Towards biologically realistic multi-compartment neuron model emulation in analog VLSI,” in *ESANN*. ESANN, 2012.

[204] G. Rovere, Q. Ning, C. Bartolozzi, and G. Indiveri, “Ultra low leakage synaptic scaling circuits for implementing homeostatic plasticity in neuromorphic architectures,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 2014, pp. 2073–2076.

[205] S. Saighi, J. Tomas, Y. Bornat, and S. Renaud, “A conductance-based silicon neuron with dynamically tunable model parameters,” in *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.* Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005., 2005, pp. 285–288.

[206] S. Saighi, Y. Bornat, J. Tomas, and S. Renaud, “Neuromimetic ICs and system for parameters extraction in biological neuron models,” in *2006 IEEE International Symposium on Circuits and Systems*, 2006, pp. 4207–4210.

[207] S. Saighi, Y. Bornat, J. Tomas, G. L. Masson, and S. Renaud, “A library of analog operators based on the hodgkin-huxley formalism for the design of tunable, real-time, silicon neurons,” *IEEE transactions on biomedical circuits and systems*, vol. 5, pp. 3–19, 2010.

[208] T. Yu and G. Cauwenberghs, “Analog VLSI neuromorphic network with programmable membrane channel kinetics,” in *2009 IEEE International Symposium on Circuits and Systems*. 2009 IEEE International Symposium on Circuits and Systems, 2009, pp. 349–352.

[209] T. Yu, T. J. Sejnowski, and G. Cauwenberghs, “Biophysical neural spiking, bursting, and excitability dynamics in reconfigurable analog VLSI,” *IEEE transactions on biomedical circuits and systems*, vol. 5, pp. 420–429, 2011.

[210] B. Marr and J. Hasler, “Compiling probabilistic, bio-inspired circuits on a field programmable analog array,” *Frontiers in neuroscience*, vol. 8, p. 86, 2014.

[211] B. McGinley, P. Rocke, F. Morgan, and J. Maher, “Reconfigurable analogue hardware evolution of adaptive spiking neural network controllers,” in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 2008, pp. 289–290.

[212] P. Rocke, B. McGinley, J. Maher, F. Morgan, and J. Harkin, “Investigating the suitability of FPAAs for evolved hardware spiking neural networks,” in *International Conference on Evolvable Systems*. International Conference on Evolvable Systems, 2008, pp. 118–129.

[213] J. Zhao and Y.-B. Kim, “Circuit implementation of Fitzhugh-Nagumo neuron model using field programmable analog arrays,” in *2007 50th Midwest Symposium on Circuits and Systems*. 2007 50th Midwest Symposium on Circuits and Systems, 2007, pp. 772–775.

[214] S. Nease, S. George, P. Hasler, S. Koziol, and S. Brink, “Modeling and implementation of voltage-mode CMOS dendrites on a reconfigurable analog platform,” *IEEE transactions on biomedical circuits and systems*, vol. 6, pp. 76–84, 2011.

[215] E. Farquhar, C. Gordon, and P. Hasler, “A field programmable neural array,” in *2006 IEEE International Symposium on Circuits and Systems*, 2006, pp. 4114–4117.

[216] M. Liu, H. Yu, and W. Wang, “Fpaa based on integration of CMOS and nanojunction devices for neuromorphic applications,” in *International Conference on Nano-Networks*. International Conference on Nano-Networks, 2008, pp. 44–48.

[217] M. R. Azghadi, S. Moradi, and G. Indiveri, “Programmable neuromorphic circuits for spike-based neural dynamics,” in *2013 IEEE 11th International New Circuits and Systems Conference (NEWCAS)*. 2013 IEEE 11th International New Circuits and Systems Conference (NEWCAS), 2013, pp. 1–4.

[218] F. Corradi, H. You, M. Giulioni, and G. Indiveri, “Decision making and perceptual bistability in spike-based neuromorphic VLSI systems,” in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2015 IEEE International Symposium on Circuits and Systems (ISCAS), 2015, pp. 2708–2711.

[219] M. A. Petrovici, B. Vogginger, P. Müller, O. Breitwieser, M. Lundqvist, L. Muller, M. Ehrlich, A. Destexhe, A. Lansner, R. Schüffny *et al.*, “Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms,” *PloS one*, vol. 9, no. 10, p. e108590, 2014.

[220] H. Djahanshahi, M. Ahmadi, G. A. Jullien, and W. C. Miller, “A unified synapse-neuron building block for hybrid VLSI neural networks,” in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, vol. 3. 1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96, 1996, pp. 483–486.

[221] S. Mitra, G. Indiveri, and S. Fusi, “Learning to classify complex patterns using a VLSI network of spiking neurons,” in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20. Curran Associates, Inc., 2008.

[222] J. V. Arthur and K. Boahen, “Learning in silicon: Timing is everything,” in *Advances in neural information processing systems*. Advances in neural information processing systems, 2006, pp. 75–82.

[223] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, “Energy-efficient neuron, synapse and STDP integrated circuits,” *IEEE transactions on biomedical circuits and systems*, vol. 6, pp. 246–256, 2012.

[224] S. Hussain, A. Basu, M. Wang, and T. J. Hamilton, “DELTRON: Neuromorphic architectures for delay based learning,” in *2012 IEEE Asia Pacific Conference on Circuits and Systems*. 2012 IEEE Asia Pacific Conference on Circuits and Systems, 2012, pp. 304–307.

[225] L. Alvado, S. Saïghi, J. Tomas, and S. Renaud, “An Exponential-Decay Synapse Integrated Circuit For Bio-inspired Neural Networks.” in *International Work-Conference on Artificial Neural Networks*. Springer, 2003, pp. 670–677.

[226] S. R. Deiss, R. J. Douglas, A. M. Whatley, and W. Maass, “A pulse-coded communications infrastructure for neuromorphic systems,” *Pulsed neural networks*, pp. 157–178, 1999.

[227] F. Sargeni and V. Bonaiuto, “An interconnection architecture for integrate and fire neuromorphic multi-chip networks,” in *2009 52nd IEEE International Midwest Symposium on Circuits and Systems*. 2009 52nd IEEE International Midwest Symposium on Circuits and Systems, 2009, pp. 877–880.

[228] A. Cassidy, T. Murray, A. G. Andreou, and J. Georgiou, “Evaluating on-chip interconnects for low operating frequency silicon neuron arrays,” in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. 2011 IEEE International Symposium of Circuits and Systems (ISCAS), 2011, pp. 2437–2440.

[229] D. Vainbrand and R. Ginosar, “Scalable network-on-chip architecture for configurable neural networks,” *Microprocessors and Microsystems*, vol. 35, pp. 152–166, 2011.

[230] S. Pande, F. Morgan, G. Smit, T. Bruintjes, J. Rutgers, B. McGinley, S. Cawley, J. Harkin, and L. McDaid, “Fixed latency on-chip interconnect for hardware spiking neural network architectures,” *Parallel computing*, vol. 39, pp. 357–371, 2013.

[231] Y. Suzuki and L. E. Atlas, “A study of regular architectures for digital implementation of neural networks,” in *IEEE International Symposium on Circuits and Systems,*. IEEE International Symposium on Circuits and Systems,, 1989, pp. 82–85.

[232] A. Mortara and E. A. Vittoz, “A communication architecture tailored for analog VLSI artificial neural networks: intrinsic performance and

limitations," *IEEE Transactions on neural networks*, vol. 5, pp. 459–466, 1994.

[233] A. Mortara, E. A. Vittoz, and P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 660–669, 1995.

[234] P. Merolla, J. Arthur, R. Alvarez, J.-M. Bussat, and K. Boahen, "A multicast tree router for multichip neuromorphic systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, pp. 820–833, 2013.

[235] P. A. Merolla, J. V. Arthur, B. E. Shi, and K. A. Boahen, "Expandable networks for neuromorphic chips," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, pp. 301–311, 2007.

[236] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-based neuromorphic systems*. John Wiley & Sons, 2014.

[237] M. Jabłoński, T. Serrano-Gotarredona, and B. Linares-Barranco, "High-speed serial interfaces for event-driven neuromorphic systems," in *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2015, pp. 1–4.

[238] S. Ramakrishnan, R. Wunderlich, J. Hasler, and S. George, "Neuron array with plastic synapses and programmable dendrites," *IEEE transactions on biomedical circuits and systems*, vol. 7, pp. 631–642, 2013.

[239] C. Zamarreño-Ramos, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco, "Multicasting mesh AER: A scalable assembly approach for reconfigurable neuromorphic structured AER systems. Application to ConvNets," *IEEE transactions on biomedical circuits and systems*, vol. 7, pp. 82–102, 2012.

[240] E. Chicca, A. M. Whatley, P. Lichtsteiner, V. Dante, T. Delbruck, P. D. Giudice, R. J. Douglas, and G. Indiveri, "A multichip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, pp. 981–993, 2007.

[241] R. Paz-Vicente, A. Linares-Barranco, D. Cascado, M. Rodriguez, G. Jimenez, A. Civit, and J. Sevillano, "PCI-AER interface for neuro-inspired spiking systems," in *2006 IEEE International Symposium on Circuits and Systems*, 2006, p. 4.

[242] V. Thanasoulis, J. Partzsch, S. Hartmann, C. Mayr, and R. Schüffny, "Dedicated fpga communication architecture and design for a large-scale neuromorphic system," in *2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012)*. 2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012), 2012, pp. 877–880.

[243] V. Thanasoulis, B. Vogginger, J. Partzsch, and R. Schüffny, "A pulse communication flow ready for accelerated neuromorphic experiments," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 2014, pp. 265–268.

[244] S. Scholze, H. Eisenreich, S. Höppner, G. Ellguth, S. Henker, M. Ander, S. Hänzsche, J. Partzsch, C. Mayr, and R. Schüffny, "A 32 GBit/s communication SoC for a waferscale neuromorphic system," *Integration*, vol. 45, pp. 61–75, 2012.

[245] T. Sharp, C. Patterson, and S. Furber, "Distributed configuration of massively-parallel simulation on Spinnaker neuromorphic hardware," in *The 2011 International Joint Conference on Neural Networks*. The 2011 International Joint Conference on Neural Networks, 2011, pp. 1099–1105.

[246] D. Brüderle, M. A. Petrovici, B. Vogginger, M. Ehrlich, T. Pfeil, S. Millner, A. Grübl, K. Wendt, E. Müller, M.-O. Schwartz *et al.*, "A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems," *Biological cybernetics*, vol. 104, pp. 263–296, 2011.

[247] F. Galluppi, S. Davies, S. Furber, T. Stewart, and C. Eliasmith, "Real time on-chip implementation of dynamical systems with spiking neurons," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–8.

[248] F. Galluppi, S. Davies, A. Rast, T. Sharp, L. A. Plana, and S. Furber, "A hierachical configuration system for a massively parallel neural hardware platform," in *Proceedings of the 9th conference on Computing Frontiers*. Proceedings of the 9th conference on Computing Frontiers, 2012, pp. 183–192.

[249] Y. Ji, Y. Zhang, S. Li, P. Chi, C. Jiang, P. Qu, Y. Xie, and W. Chen, "NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2016, pp. 1–13.

[250] G. Urgese, F. Barchi, E. Macii, and A. Acquaviva, "Optimizing network traffic for spiking neural network simulations on densely interconnected many-core neuromorphic platforms," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, pp. 317–329, 2016.

[251] Q. X. Wu, X. Liao, X. Huang, R. Cai, J. Cai, and J. Liu, "Development of FPGA toolbox for implementation of spiking neural networks," in *2015 Fifth International Conference on Communication Systems and Network Technologies*. 2015 Fifth International Conference on Communication Systems and Network Technologies, 2015, pp. 806–810.

[252] A. D. Brown, S. B. Furber, J. S. Reeve, J. D. Garside, K. J. Dugan, L. A. Plana, and S. Temple, "SpiNNaker—programming model," *IEEE Transactions on Computers*, vol. 64, pp. 1769–1782, 2014.

[253] A. Disney, J. Reynolds, C. D. Schuman, A. Klibisz, A. Young, and J. S. Plank, "Danna: A neuromorphic software ecosystem," *Biologically Inspired Cognitive Architectures*, vol. 17, pp. 49–56, 2016.

[254] N. Kasabov, N. M. Scott, E. Tu, S. Marks, N. Sengupta, E. Capecci, M. Othman, M. G. Doborjeh, N. Murli, R. Hartono *et al.*, "Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: design methodology and selected applications," *Neural Networks*, vol. 78, pp. 1–14, 2016.

[255] N. Scott, N. Kasabov, and G. Indiveri, "Neucube neuromorphic framework for spatio-temporal brain data and its python implementation," in *International Conference on Neural Information Processing*. International Conference on Neural Information Processing, 2013, pp. 78–84.

[256] Y. Ji, Y.-H. Zhang, and W.-M. Zheng, "Modelling spiking neural network from the architecture evaluation perspective," *Journal of computer science and technology*, vol. 31, pp. 50–59, 2016.

[257] M. Plagge, C. D. Carothers, E. Gonsiorowski, and N. Mcglohon, "Nemo: A massively parallel discrete-event simulation model for neuromorphic architectures," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 28, pp. 1–25, 2018.

[258] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, "Compass: A scalable simulator for an architecture for cognitive computing," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–11.

[259] L. Xia, B. Li, T. Tang, P. Gu, P.-Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, "Mnsim: Simulation platform for memristor-based neuromorphic computing system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 1009–1022, 2017.

[260] Q. Chen, R. Luley, Q. Wu, M. Bishop, R. W. Linderman, and Q. Qiu, "Anrad: A neuromorphic anomaly detection framework for massive concurrent data streams," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1622–1636, 2017.

[261] G. Haessig, A. Cassidy, R. Alvarez, R. Benosman, and G. Orchard, "Spiking optical flow for event-based sensors using ibm's truenorth neurosynaptic system," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 4, pp. 860–870, 2018.

[262] M. Z. Alom, B. Van Essen, A. T. Moody, D. P. Widemann, and T. M. Taha, "Quadratic unconstrained binary optimization (QUBO) on neuromorphic computing system," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3922–3929.

[263] E. Nurse, B. S. Mashford, A. J. Yepes, I. Kiral-Kornek, S. Harrer, and D. R. Freestone, "Decoding eeg and lfp signals using deep learning: heading truenorth," in *Proceedings of the ACM international conference on computing frontiers*, 2016, pp. 259–266.

[264] S. Moran, B. Gaonkar, W. Whitehead, A. Wolk, L. Macyszyn, and S. S. Iyer, "Deep learning for medical image segmentation–using the IBM truenorth neurosynaptic system," in *Medical Imaging 2018: Imaging Informatics for Healthcare, Research, and Applications*, vol. 10579. International Society for Optics and Photonics, 2018, p. 1057915.

[265] J.-P. Pfister and W. Gerstner, "Triplets of spikes in a model of spike timing-dependent plasticity," *Journal of Neuroscience*, vol. 26, no. 38, pp. 9673–9682, 2006.

[266] J. Gjorgjieva, C. Clopath, J. Audet, and J.-P. Pfister, "A triplet spike-timing–dependent plasticity model generalizes the Bienenstock–Cooper–Munro rule to higher-order spatiotemporal correlations," *Proceedings of the National Academy of Sciences*, vol. 108, no. 48, pp. 19 383–19 388, 2011.

[267] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural computation*, vol. 19, no. 6, pp. 1468–1502, 2007.

[268] C. Yakopcic, N. Rahman, T. Atahary, T. M. Taha, and S. Douglass, "Solving constraint satisfaction problems using the loihi spiking neuromorphic processor," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1079–1084.

[269] E. P. Frady, G. Orchard, D. Florey, N. Imam, R. Liu, J. Mishra, J. Tse, A. Wild, F. T. Sommer, and M. Davies, "Neuromorphic Nearest Neighbor Search Using Intel's Pohoiki Springs," in *Proceedings of the Neuro-inspired Computational Elements Workshop*, 2020, pp. 1–10.

[270] W. Severa, R. Lehoucq, O. Parekh, and J. B. Aimone, "Spiking neural algorithms for markov process random walk," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[271] N. Imam and T. A. Cleland, "Rapid online learning and robust recall in a neuromorphic olfactory circuit," *Nature Machine Intelligence*, vol. 2, no. 3, pp. 181–191, 2020.

[272] M. Evanusa, Y. Sandamirskaya *et al.*, "Event-based attention and tracking on neuromorphic hardware," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[273] C. D. Schuman, J. P. Mitchell, R. M. Patton, T. E. Potok, and J. S. Plank, "Evolutionary optimization for neuromorphic systems," in *Proceedings of the Neuro-inspired Computational Elements Workshop*, 2020, pp. 1–9.

[274] D. Rasmussen, "Nengodl: Combining deep learning and neuromorphic modelling methods," *Neuroinformatics*, vol. 17, no. 4, pp. 611–628, 2019.

[275] T. DeWolf, P. Jaworski, and C. Eliasmith, "Nengo and low-power ai hardware for robust, embedded neurorobotics," *Frontiers in Neurorobotics*, vol. 14, 2020.

[276] K. Amunts, C. Ebell, J. Muller, M. Telefont, A. Knoll, and T. Lippert, "The human brain project: creating a european research infrastructure to decode the human brain," *Neuron*, vol. 92, pp. 574–581, 2016.

[277] E. DAngelo, G. Danese, G. Florimbi, F. Leporati, A. Majani, S. Masoli, S. Solinas, and E. Torti, "The human brain project: High performance computing for brain cells hw/sw simulation and understanding," in *2015 Euromicro Conference on Digital System Design*. 2015 Euromicro Conference on Digital System Design, 2015, pp. 740–747.

[278] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of neurophysiology*, vol. 94, pp. 3637–3642, 2005.

[279] S. Millner, "Development of a multi-compartment neuron model emulation," Ph.D. dissertation, 2012.

[280] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010, pp. 1947–1950.

[281] E. Müller, S. Schmitt, B. Vogginger, D. Lester, and T. P. A. P. Davison. (2020, Jan.) HBP Neuromorphic Computing Platform Guidebook. [Online]. Available: https://electronicvisions.github.io/hbp-sp9-guidebook/

[282] G. Indiveri, "Neuromorphic VLSI models of selective attention: from single chip vision sensors to multi-chip systems," *Sensors*, vol. 8, pp. 5352–5375, 2008.

[283] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 431–438.

[284] D. Brüderle, E. Müller, A. P. Davison, E. Muller, J. Schemmel, and K. Meier, "Establishing a novel modeling tool: a python-based interface for a neuromorphic hardware system," *Frontiers in neuroinformatics*, vol. 3, p. 17, 2009.

[285] C. S. Thakur, J. L. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. O. Hasler *et al.*, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in neuroscience*, vol. 12, p. 891, 2018.

[286] S. Schmitt, J. Klähn, G. Bellec, A. Grübl, M. Guettler, A. Hartel, S. Hartmann, D. Husmann, K. Husmann, S. Jeltsch *et al.*, "Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system," in *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2227–2234.

[287] L. A. Plana, S. B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, and S. Yang, "A gals infrastructure for a massively parallel multiprocessor," *IEEE Design & Test of Computers*, vol. 24, pp. 454–463, 2007.

[288] J. van Albada, Sacha, A. G. Rowley, J. Senk, M. Hopkins, M. Schmidt, A. B. Stokes, D. R. Lester, M. Diesmann, and S. B. Furber, "Performance comparison of the digital neuromorphic hardware Spinnaker and the neural network simulation software nest for a full-scale cortical microcircuit model," *Frontiers in neuroscience*, vol. 12, p. 291, 2018.

[289] E. I. Guerra-Hernandez, A. Espinal, P. Batres-Mendoza, C. H. Garcia-Capulin, R. D. J. Romero-Troncoso, and H. Rostro-Gonzalez, "A FPGA-based neuromorphic locomotion system for multi-legged robots," *IEEE Access*, vol. 5, pp. 8301–8312, 2017.

[290] O. Rhodes, P. A. Bogdan, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D. R. Lester, M. Mikaitis, L. A. Plana, A. G. D. Rowley *et al.*, "sPyNNaker: a software package for running Pynn simulations on SpiNNaker," *Frontiers in neuroscience*, vol. 12, p. 816, 2018.

[291] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "PyNN: a common interface for neuronal network simulators," *Frontiers in neuroinformatics*, vol. 2, p. 11, 2009.

[292] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in neuroinformatics*, vol. 7, p. 48, 2014.

[293] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, p. 1430, 2007.

[294] M. Stimberg, R. Brette, and D. F. M. Goodman, "Brian 2, an intuitive and efficient neural simulator," *Elife*, vol. 8, 2019.

[295] M. Stimberg, D. F. M. Goodman, V. Benichoux, and R. Brette, "Equation-oriented specification of neural models for simulations," *Frontiers in neuroinformatics*, vol. 8, p. 6, 2014.

[296] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He *et al.*, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[297] Y. Mi, C. C. A. Fung, K. Y. M. Wong, and S. Wu, "Spike frequency adaptation implements anticipative tracking in continuous attractor neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014, pp. 505–513.

[298] S. Furber, "Large-scale neuromorphic computing systems," *Journal of neural engineering*, vol. 13, no. 5, p. 051001, 2016.

[299] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, pp. 507–519, 1971.

[300] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, pp. 80–83, 2008.

[301] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Letters*, vol. 10, pp. 1297–1301, 2010.

[302] R. Islam, H. Li, P.-Y. Chen, W. Wan, H.-Y. Chen, B. Gao, H. Wu, S. Yu, K. Saraswat, and H.-S. P. Wong, "Device and materials requirements for neuromorphic computing," *Journal of Physics D*, vol. 52, p. 113001, 2019.

[303] P.-Y. Chen, "Design of Resistive Synaptic Devices and Array Architectures for Neuromorphic Computing," Ph.D. dissertation, Arizona State University, 2018.

[304] P.-Y. Chen, J.-s. Seo, Y. Cao, and S. Yu, "Compact oscillation neuron exploiting metal-insulator-transition for neuromorphic computing," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2016, pp. 1–6.

[305] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, M. Hu, N. Ge, J. P. Strachan, Z. Li *et al.*, "Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing," *Nature materials*, vol. 16, pp. 101–108, 2017.

[306] Z. Wang, S. Joshi, S. Savel'ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo, H. Jiang, P. Lin, C. Li, J. H. Yoon, N. K. Upadhyay, J. Zhang, M. Hu, J. P. Strachan, M. Barnell, Q. Wu, H. Wu, R. S. Williams, Q. Xia, and J. J. Yang, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electronics*, vol. 1, pp. 137–145, 2018.

[307] Z. Q. Wang, H. Y. Xu, X. H. Li, H. Yu, Y. C. Liu, and X. J. Zhu, "Synaptic learning and memory functions achieved using oxygen ion migration/diffusion in an amorphous ingazno memristor," *Advanced Functional Materials*, vol. 22, pp. 2759–2765, 2012.

[308] M. R. Azghadi, B. Linares-Barranco, D. Abbott, and P. H. W. Leong, "A hybrid CMOS-memristor neuromorphic synapse," *IEEE transactions on biomedical circuits and systems*, vol. 11, pp. 434–445, 2016.

[309] G. C. Adam, B. D. Hoskins, M. Prezioso, F. Merrikh-Bayat, B. Chakrabarti, and D. B. Strukov, "3-d memristor crossbars for

analog and neuromorphic computing applications," *IEEE Transactions on Electron Devices*, vol. 64, pp. 312–318, 2017.

[310] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, and J. P. Strachan, "Memristor-Based Analog Computation and Neural Network Classification with a Dot Product Engine," *Advanced Materials*, vol. 30, p. 1705914, 2018.

[311] Y. Zhang, Z. Wang, J. Zhu, Y. Yang, M. Rao, W. Song, Y. Zhuo, X. Zhang, M. Cui, L. Shen, R. Huang, and J. J. Yang, "Brain-inspired computing with memristors: Challenges in devices, circuits, and systems," *Applied physics reviews*, vol. 7, p. 11308, 2020.

[312] B. Yan, A. M. Mahmoud, J. J. Yang, Q. Wu, Y. Chen, and H. H. Li, "A neuromorphic ASIC design using one-selector-one-memristor crossbar," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 1390–1393.

[313] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, 2017, pp. 19–24.

[314] M. Hu, H. Li, Q. Wu, G. S. Rose, and Y. Chen, "Memristor crossbar based hardware realization of BSB recall function," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–7.

[315] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd Annual Design Automation Conference on*. Proceedings of the 53rd Annual Design Automation Conference on, 2016, p. 19.

[316] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nature communications*, vol. 4, pp. 1–7, 2013.

[317] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems," *Frontiers in neuroscience*, vol. 7, p. 2, 2013.

[318] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, pp. 61–64, 2015.

[319] X. Zhu, C. Du, Y. Jeong, and W. D. Lu, "Emulation of synaptic metaplasticity in memristors," *Nanoscale*, vol. 9, pp. 45–51, 2017.

[320] C. Zamarreño-Ramos, L. A. Camuñas-Mesa, J. A. Pérez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex." *Frontiers in Neuroscience*, vol. 5, pp. 26–26, 2011.

[321] D. Negrov, I. Karandashev, V. Shakirov, Y. Matveyev, W. Dunin-Barkowski, and A. Zenkevich, "An approximate backpropagation learning rule for memristor based neural networks using synaptic plasticity," *Neurocomputing*, vol. 237, pp. 193–199, 2017.

[322] R. Guo, Y. Zhou, L. Wu, Z. Wang, Z. Lim, X. Yan, W. Lin, H. Wang, H. Y. Yoong, S. Chen, Ariando, T. Venkatesan, J. Wang, G. M. Chow, A. Gruverman, X. Miao, Y. Zhu, and J. Chen, "Control of Synaptic Plasticity Learning of Ferroelectric Tunnel Memristor by Nanoscale Interface Engineering," *ACS Applied Materials & Interfaces*, vol. 10, pp. 12 862–12 869, 2018.

[323] Z. I. Mannan, S. P. Adhikari, C. Yang, R. K. Budhathoki, H. Kim, and L. Chua, "Memristive imitation of synaptic transmission and plasticity," *IEEE Transactions on Neural Networks*, vol. 30, pp. 3458–3470, 2019.

[324] R. Yang, H. Huang, Q. Hong, X. Yin, Z. Tan, T. Shi, Y. Zhou, X. Miao, X. Wang, S. Mi, C. Jia, and X. Guo, "Synaptic suppression Triplet-STDP learning rule realized in second-order memristors," *Advanced Functional Materials*, vol. 28, p. 1704455, 2018.

[325] Z. Cheng, C. Ríos, W. H. P. Pernice, C. D. Wright, and H. Bhaskaran, "On-chip photonic synapse," *Science Advances*, vol. 3, 2017.

[326] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice, "All-optical spiking neurosynaptic networks with self-learning capabilities," *Nature*, vol. 569, pp. 208–214, 2019.

[327] A. Hurtado and J. Javaloyes, "Controllable spiking patterns in long-wavelength vertical cavity surface emitting lasers for neuromorphic photonics systems," *Applied Physics Letters*, vol. 107, p. 241103, 2015.

[328] B. Gholipour, P. Bastock, C. Craig, K. Khan, D. Hewak, and C. Soci, "Amorphous Metal-Sulphide Microfibers Enable Photonic Synapses for Brain-Like Computing," *Advanced Optical Materials*, vol. 3, pp. 635–641, 2015.

[329] A. N. Tait, T. F. D. Lima, M. A. Nahmias, H. B. Miller, H. T. Peng, B. J. Shastri, and P. R. Prucnal, "Silicon Photonic Modulator Neuron," *Physical review applied*, vol. 11, p. 64043, 2019.

[330] A. N. Tait, T. F. D. Lima, E. Zhou, A. X. Wu, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Neuromorphic photonic networks using silicon photonic weight banks," *Scientific reports*, vol. 7, p. 1–10, 2017.

[331] A. N. Tait, M. A. Nahmias, Y. Tian, B. J. Shastri, and P. R. Prucnal, "Photonic neuromorphic signal processing and computing," in *Nanophotonic Information Physics*. Springer, 2014, pp. 183–222.

[332] M. Lee, W. Lee, S. Choi, J.-W. Jo, J. Kim, S. K. Park, and Y.-H. Kim, "Brain-inspired Photonic Neuromorphic Devices using Photodynamic Amorphous Oxide Semiconductors and their Persistent Photoconductivity," *Advanced Materials*, vol. 29, p. 1700951, 2017.

[333] Y. Wang, Z. Lv, J. Chen, Z. Wang, Y. Zhou, L. Zhou, X. Chen, and S. Han, "Photonic synapses based on inorganic perovskite quantum dots for neuromorphic computing," *Advanced Materials*, vol. 30, p. 1802883, 2018.

[334] A. Katumba, M. Freiberger, F. Laporte, A. Lugnan, S. Sackesyn, C. Ma, J. Dambre, and P. Bienstman, "Neuromorphic computing based on silicon photonics and reservoir computing," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 24, pp. 1–10, 2018.

[335] M. Miscuglio, A. Mehrabian, Z. Hu, S. I. Azzam, J. George, A. V. Kildishev, M. Pelton, and V. J. Sorger, "All-optical nonlinear activation function for photonic neural networks [invited]," *Optical Materials Express*, vol. 8, pp. 3851–3863, 2018.

[336] I. Chakraborty, G. Saha, A. Sengupta, and K. Roy, "Toward Fast Neural Computing using All-Photonic Phase Change Spiking Neurons," *Scientific Reports*, vol. 8, p. 12980, 2018.

[337] H.-T. Peng, M. A. Nahmias, T. de Lima, Ferreira, A. N. Tait, and B. J. Shastri, "Neuromorphic Photonic Integrated Circuits," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 24, pp. 1–15, 2018.