IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier

A Survey of Intelligent Chip Design Research Based on Spiking Neural Networks

LU CHEN[1,2], XINGZHONG XIONG[1,2], and JUN LIU[1,2]
[1]School of Automation and Information Engineering, Sichuan University of Science and Engineering, Yibin 644000, China
[2] Artificial Intelligence Key Laboratory of Sichuan Province, Yibin 644000, China

Corresponding author:Xingzhong Xiong (e-mail: xzxiong@suse.edu.cn).

This work was supported in part by the Sichuan Science and Technology Program under Grant 2021YFG0127, and in part by the 2021 Graduate Innovation Fund of Sichuan University of Science and Engineering under Grant y2021086.

ABSTRACT The traditional neural network Intelligent chip has the problem of high power consumption due to classical computing architecture, limiting the development of neural network Intelligent chips. Stochastic computing (SC) encodes binary numbers into stochastic pulse sequences in operation, taking advantage of low power consumption and high performance. The application of SC in spiking neural networks (SNNs) Intelligent chips is beneficial to solving the high power consumption of traditional neural network chips. This article first summarizes the basic elements of SNNs and the basic principles of SC. Then, we review the development trends of the stochastic computation-based neural network chips and existing SNN chips under research at home and abroad, respectively, and analyze the current problems. Finally, a review of SNN chips based on SC is highlighted. This paper aims to provide new research directions and to learn ideas for the field of SNN chips through systematic summaries.

INDEX TERMS Spiking Neural Network; Spiking Neuron Model; Neural Computing; Stochastic Computing; Brain-like Chip

I. INTRODUCTION

THE rapid development of computer hardware has promoted the development of deep learning, which has been developed to date and has made great achievements in autonomous driving [1], pattern recognition [2], data classification [3], etc. However, the current stage of deep learning has hindered the further development of artificial intelligence due to its high power consumption, long training time, and low brightness [4]. Unlike the traditional artificial neural networks (ANNs), biological neural networks communicate through discrete pulses rather than numerical values to form SNNs. In SNNs, neurons are activated only when input pulses are received. Thus inactive neurons without input pulses can be placed in low-power mode, thereby reducing power consumption and simplifying computation. As a result, SNNs can potentially achieve extremely low power consumption compared to ANNs, especially when implementing analog/mixed-signal (AMS) circuits. In addition, spiking neural network-based brain-like computing is a better way to overcome the shortcomings of the current deep learning stage and solve artificial intelligence problems because the working mechanism is closer to that of the biological brain [5-7].

The implementation methods of SNNs can be divided into software and hardware implementation. Software simulation can quickly realize neuron modelling and real-time data analysis of network communication systems; it plays a vital role in numerical processing and optimization. Hardware implementation focuses more on the design and implementation of neuromorphic hardware architectures. Studies have shown that although the software implementation has the characteristics of solid flexibility and high precision [8], it cannot fully use the high parallelism of neural networks, the processing speed is slow, and power consumption is high. Hardware implementation can improve the deficiencies of software implementation, fully reflecting the characteristics of high parallelism of the neural network [9].

According to different hardware implementations, neural network chips can be divided into analog, digital, and digital-analog hybrids [10]. Analog circuits have higher computational accuracy, but due to the complexity of their design, the scale of neural networks is usually small. The factors also cannot guarantee the consistency of neural network behavior between chips, so pure simulation methods are rarely used to build large-scale brain-like systems. Most large brain computers that have emerged are digital or mixed digital-analog. However, the current general-purpose general-purpose CPU,GPU, and TPU chips have fatal problems: first, the convolutional computation of too much data causes a surge in power consumption of traditional AI chips, which is not conducive to edge-side deployment of AI, i.e., the "power wall" problem; second, the deep network contains a large number of neural networks. Second, because the deep network includes many weight parameters, it creates higher requirements for bandwidth and latency, resulting in a computational bottleneck in the whole system, i.e., the "memory wall" problem [11].

As an unconventional computing paradigm, stochastic computing is one of the important implementations of neural networks. Its low-cost and low-loss circuitry can perform the functions of complex circuits, with lower hardware overhead and better fault tolerance, etc., has attracted attention. The stochastic computing uses discrete pulse sequences instead of sequential binary numbers, occupying lower computational resources. At the same time, it also pays the price of higher computational latency and lower computational accuracy. Some researchers have already made some preliminary attempts to address the above issues. Brown, Card et al. used finite state machine (FSM) processes to implement nonlinear functions and thus improve the computational accuracy [12].

Based on this idea, Smithson et al. proposed using FSM processes to implement the pulse issuing process of leaky integrate and fire (LIF) neurons and its hardware architecture [13]; the deterministic bit stream proposed by Faraji can significantly reduce energy consumption. Without changing the network performance [14], Sim et al. proposed an improved probabilistic coding method to optimize computational latency and computational accuracy [15]. Kim et al. proposed using SC in deep neural networks for dynamic energy-accuracy trade-offs to improve computational hardware efficiency [16]. Liu et al. proposed an energy-efficient Deep Belief Network (DBN) based on the online learning ability of SC and improved its computational efficiency by improving random numbers [17]. Riedel et al. proposed a SC method for deterministic bitstreams, which can achieve better computational accuracy [18]. In 2020, Huang's team investigated the reliability of stochastic logic circuits based on Fin Field-Effect Transistor (FinFET) technology, which provides a good prospect for the application of new nanodevices [19].

However, building neural network computing architectures using SC methods is still challenging. SC is used in the design of SNNs chips to construct more efficient parallel computational processing units based on the characteristics of SC and to improve computational efficiency to enhance the performance of SNNs gas pedals. The coding method of SC is enhanced so that the stochastic computation-based spiking neural network chip has the advantages of both low power consumption and high efficiency.

In this paper, from the characteristics of SNNs, such as good bionic properties, high efficiency and low power consumption, we review the neuron model, network topology, and learning algorithm for SNN-like brain neural chips in order. In section 2, the basic elements and biological background of SNN and the basic principles of SC are described. Section 3 presents the software optimization of SNNs and the communication protocol of the SNN accelerator. Section 4 introduces the research progress of three traditional SNN chips, and summarizes the existing SNN chips. Section 5 focuses on the application of SC in traditional neural network chips and SNN chips. Section 6 provides an outlook on the future development of SNN chips. Section 7 summarize the work of this paper.

II. BASIC ELEMENTS OF SNN AND STOCHASTIC COMPUTING

A. SPIKING NEURAL NETWORK

The ANN is an abstraction and simulation of the structure and function of the biological nervous system and plays an important role in information processing and pattern recognition. SNNs are special ANNs, also known as third-generation ANNs, in which neuron units communicate using discrete spike trains, as shown in Figure 1. Similar to biological neurons, the input of a spiking neuron is a discrete spike, and only when the input exceeds a certain threshold will a pulse be released to the next neuron. SNNs also incorporate temporal dynamics, which makes them suitable for real-time operation with the event- and data-driven updates.

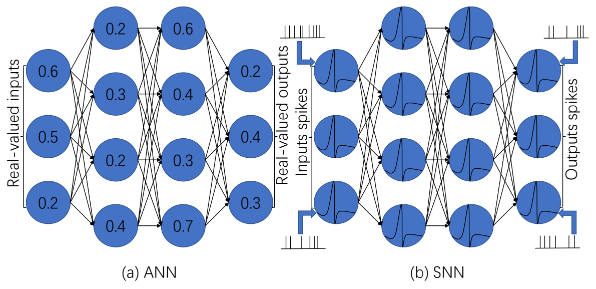


FIGURE 1. Schematic representation of ANN and SNN.

Figure 1(a) depicts a typical ANN with artificial neurons as the computing unit. A continuous function is used as the neuron's activation function to realize real value's input and output processing. The calculation process can be described by the formula 1 as follows:

$$y=\phi(\mathrm{b}+\sum_j x_j\omega_j) \tag{1}$$

Where x, y, ω, and b represent the input function, output function, salience weight, and bias, j is the input neuron index, and $\phi(.)$means the activation function. Neurons in an ANN communicate using high-precision, and continuous-value encoded and propagate information layer by layer only in the spatial domain [20].

Figure 1(b) depicts the SNN with spiking neurons as the computing unit. Compared to ANN neurons, which have a similar structure but behave differently, spiking neurons communicate information through binary time-coded spike trains. Dendrites integrate input spikes, unlike continuous activation in ANNs. A spiking neuron can be described as follows [21]:

$$\begin{cases} \frac{dX}{dt} = f(x) \\ \text{X} \leftarrow g_i(X) \end{cases} \tag{2}$$

Where X represents the vector consisting of the state variables of the neuron, $f(.)$ represents the differential equation for the evolution of the state variables, and $g_i(.)$ represents the change of the state variables caused by the spike events of synapse i. Neural information in spiking neurons is transmitted and processed by precisely timed spike trains. Compared with the ANN model, SNN can describe the real biological nervous system more accurately, thus realizing efficient information processing.

1) NEURON MODEL

The pulse neuron is the basic unit that constitutes the SNN, and its main function is to integrate and transmit the coded information of the pulse sequence. Whether a pulse neuron releases a pulse or not is closely related to the neuron's membrane potential and activation threshold. In the process of a single spiking neuron firing a pulse in a spiking neural network, a spiking neuron receives input pulses from several dendrites and outputs axons from it. Many neurons form a network and learn systematically [22], as shown in figure 2.

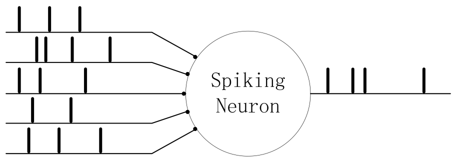


FIGURE 2. A spiking neuron receives spikes from multiple inputs, and processes and generates.

Five common spiking neuron models are the Hodgkin-Huxley (H-H) model [23 25], the leaky integrate and fire (LIF) model [26 28], and the Izhikevich model. [29][30], the integral and fire (IF) model [31] and the spike response (SRM) model [32], as shown in Figure 3.

The H-H model is a biologically interpretable physiological model. It describes the change process of the membrane potential of neurons through the dynamic characteristics of Na ion and K ion channels. The H-H model is very complex and is the most widely used model at present, and is a simplification of the H-H model. It achieves a good balance between complexity and computational accuracy. The higher the frequency of external stimulation of the LIF neuron model, the larger its activation probability [33]. The LIF neuron unifies the expressions of action potentials. It reduces the complexity of operations, but it cannot explain the fundamental pulse generation mechanism and does not include rich behavioural properties, so the LIF neuron model can only simulate a small number of neuronal behaviours [34][35][36]. The IZH neuron model has high computational efficiency and physiological characteristics similar to the H-H model. Different neuron firing patterns can be simulated through the selection of parameters [37][38]. Therefore, the IZH neuron model is implemented in some digital-analog hybrid neuromorphic systems through analog circuits, but it is not widely used because the equations are still complex. The IF model can only passively accept the external current input. After the circuit has experienced a long time, the charge of the capacitor will be released. Moreover, the IF model does not have any reset method. The IF model already possesses some of the physiological characteristics of neurons and has also been widely used in neurocomputing science. However, through many simulation experiments, it will be found that the IF neurons cannot meet the real neuron charging and discharging requirements, so the model is not perfect. The SRM model is a generalization of the IF model, so its simulated biological properties are improved. Due to the extensive selection of kernel functions, the SRM model has a certain generality compared with the LIF model. However, the SRM model is too simple to simulate many neuron charging and discharging characteristics and has certain limitations. A summary of the five common spiking neuron models is shown in table 1.

2) ENCODING METHOD

In addition to the difference in neurons, the most significant difference between SNNs and traditional neural networks is information encoding and processing. In the SNN, the input and output of data are in the form of pulse sequences, so it is necessary to convert the analog quantity into a pulse sequence with time information. The most widely used coding methods are time coding and frequency coding.

Temporal encoding focuses on differences in temporal structure, the time from receiving a stimulus to sending the first pulse, and the temporal logic between pulses containing important information. VanRullen et al. proposed the Time-to-First-spike method [39], which uses the time when a neuron first fires a spike to represent information, emphasizing the time of the first spike and ignoring other spike times. Or reduce the weight. Chien et al. proposed to use Inter spike Interval (ISI) to encode activation strength [40].

The frequency coding is mainly based on the frequency of pulse firing, the average number of pulses fired by the neuron over its corresponding recording time. Because the frequency of neuron firing pulses is positively correlated with the intensity of external stimulation, the intensity of stimulation can be expressed by the frequency of neuron firing pulses. Strong stimulation will lead to high-frequency pulse trains, and weak stimulation will lead to low-frequency pulses sequence. Frequency coding only pays attention to

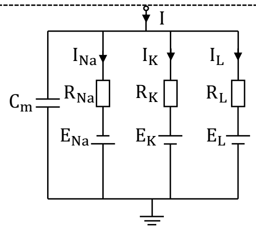
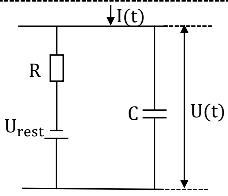
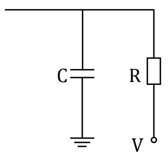
Neuron	Hodgkin-Huxley model	leaky integrate and fire model	Integrate and fire model	Izhikevich model	Spike response model
Equivalent circuit	 (a) Equivalent circuit of the H-H neuron model	 (a) Equivalent circuit of the LIF neuron model	 (c) Equivalent circuit of the IF neuron model	N/A	N/A
Equivalent equations	$C_m\frac{dV}{dt} = -g_L(V-E_L) - \overline{g_{Na}}m^3h(V-E_{Na}) - \overline{g_K}n^4(V-E_K) + I$ $\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V)m$ $\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V)h$ $\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)n$	$\tau_m\frac{dU}{dt} = -(U-U_{rest}) + RI$	$I(t) = C\frac{dV}{dt}$	$\frac{dV}{dt} = 0.04V^2 + 5V + 140 - U + I$ $\frac{dU}{dt} = a(bV - U)$	$u_i(t) = \sum_{t_i^f \in F_i}\eta_i(t-t_i^f) + \sum_{j\in\Gamma_i}\sum_{t_i^f\in F_i}\omega_{ij}\varepsilon_{ij}(t-t_i^f)$
Brief description	Where I represents input current; $C_m$refers to capacitance; V represents membrane voltage; R_{Na}、R_K and R_L represent the resistance on the ion channel; E_{Na}、E_K and E_L represent the reverse potential constant; m and n controls Na channel, Variable $h$controls channel K; α and β represent empirical function of membrane potential.	Where $\tau_m = RC$ represents the membrane time constant of neurons; I represent the sum of current; U represents the voltage at both ends of the capacitor; U_{rest} representsthe resting voltage.	I represents the current flowing through resistance R, V represents the voltage at both ends of capacitor C.	V represents the membrane voltage, U represents the auxiliary variable, a、b、c and dare model parameters, When $V \geq 30mV$, then $V \leftarrow c$, $U \leftarrow U + d$.	Γ_i represents the presynaptic set connected to neuron i; ϑ Indicates threshold voltage, Time set represented by F_i; η_i indicates reset voltage; ε_{ij} indicates the response of the pulse, τ is a time constant.

FIGURE 3. Comparison of five commonly used spiking neurons in terms of neuron equivalent circuits and equivalent equations.

TABLE 1. Summary of five commonly used spiking neuron models

Neuron model	Circuit form	Mathematical expressions	Advantages	Disadvantages	Usage Scenarios
H-H	Capacitor-resistor circuit	Differential	Close to biological neurons, high accuracy	Complex terms and cumbersome arithmetic	For the simulation of single neurons or small-scale neural networks, not for the construction of large-scale SNNs
LIF	Capacitor-resistor circuit	Differential	Simulates resting state, simple arithmetic	Does not explain the real pulse generation mechanism and does not include rich behavioural properties	Only a small amount of neuronal behavior can be simulated
IZH		Differential	Simulation of multiple discharge modes	Complex calculations and low computing efficiency	Bionic brain-computer interface and large-scale biological neuron simulation
IF	Capacitance	Differential	Simple operation	A simple model with memory effect and less than perfect model	Difficult to meet the filing requirements of real neurons
SRM	N/A	Points	Simulation of the dynamic changes of the non-period	The model is simple and can only simulate the charging and discharging characteristics of some neurons	N/A

the number of pulses in the time window while ignoring ISI. It cannot full use the temporal and spatial information contained in the pulse sequence, so the efficiency is not high [41]. Still, the non-uniqueness of the pulse sequence makes the frequency coding highly effective. Noise immunity. Time coding and frequency coding are compared, as shown in figure 4.

3) NETWORK TOPOLOGY

The topology of the SNN directly reflects the connection between neurons and synapses. The existing structure of a SNN can be divided into static and dynamic structures according to whether the network changes. The static structure means that the number of neurons and layers of the SNN remains unchanged, and only parameters such as weights are changed during the training process. Common structures include multi-layer feedforward and recurrent network structures [42]. Dynamic structure refers to the dynamic adjustment of the number and connection of neurons during the training process, typically represented by evolutionary spiking neural networks. The construction idea of the evolutionary SNN comes from the connected evolutionary system of biology, which can dynamically change the structure and function of the system in an adaptive, self-organizing, and online continuous manner. The input samples are encoded, converted into spike sequences and passed into the network.

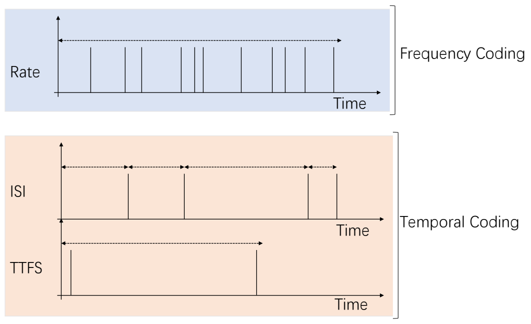


FIGURE 4. Temporal diagram of the number of emitted spikes as a function of the type of coding employed.

Then, according to the samples' characteristics, the spike network's evolutionary structure can dynamically generate new neurons and add them to the corresponding neuron reserve category. Therefore, the order rule is used in the evolutionary SNN to learn and represent the output category. The earliest activated neuron represents the corresponding category [43].

4) SNN LEARNING ALGORITHM

As the core model of brain-like computing, the learning algorithm of SNNs has always been the research focus. Studying SNN learning algorithms is beneficial for realizing higher-level artificial intelligence. With the continuous development of the field of neuromorphic chips, in addition to the performance indicators of traditional learning algorithms, indicators such as algorithm storage resource usage and weight update logic complexity are also used to measure the performance of SNN learning algorithms implemented in hardware. Because neuromorphic chips have limited computing and storage resources, there are many constraints on the algorithm to complete low-power dedicated computing modes. At present, SNN algorithms can be divided into unsupervised learning, supervised learning, online learning, network conversion, and network compression algorithms.

Unsupervised learning algorithms usually only need local information to adjust the synaptic quality during the learning process and are usually implemented in hardware. Supervised SNN learning algorithms are usually based on gradient descent algorithms to directly learn input patterns' labels. Due to the discontinuity and non-differentiability of spike trains, it is difficult for such algorithms to be extended to deep layers and effectively applied to complex datasets. Zhang et al. proposed the SpiKL-IP algorithm based on strict information theory to minimize the value of KL divergence between the actual and expected pulse firing frequency to learn the input pulse pattern in real-time. The algorithm achieved the highest recognition coefficients in the CityScape image dataset and TI46 speech corpus, reaching 97.78% and 98.46%, respectively [44]. Building on this work, Kasabov proposed a dynamic eSNN to learn information from more complex spatiotemporal input patterns consisting of multiple spikes. They added learning rules for synaptic plasticity to update the weights and used the EGG dataset. 83.33% recognition accuracy was achieved on [45]. Wysoski et al. proposed an evolutionary spiking neural network eSNN based on hierarchical sorting learning. It uses a single-pulse hierarchical time coding algorithm to encode data. It can continuously change its structure in the real-time learning process and better respond to different input modes. The recognition performance on the visual and audio datasets reaches 60% and 40%, respectively [46].

Online learning algorithms mainly refer to algorithms that can learn the information flow of external input in real-time; the online learning algorithm of SNNs with the dynamic adaptive structure proposed by Wang et al. The classification accuracy of the dataset reached 91.8% [47]. Courbariaux introduces a novel weight binarization scheme in forwarding and backward propagation. It can reduce the multipliers by $2/3$ and is three times faster when training; this method is very effective for neural networks. The hardware implementation greatly impacts the classification accuracy of 91.35% on the CIFAR-10 network [48].

The network conversion algorithm learns from the ANN, whose training algorithm is already very mature, and quickly obtains an SNN with good performance. Cao et al. describe a method for converting a convolutional neural networks (CNN) architecture to an SNN architecture that can be directly mapped to certain spiking-based neuromorphic hardware with a little performance penalty. In software, the classification accuracy rate of this architecture on the CIFAR-10 image dataset reached 77.43% [49]. The network compression algorithm mainly reduces the network size through structure pruning and weight quantization operations to facilitate hardware implementation. Rueckauer et al. address some important shortcomings of the existing ANN-to-SNN conversion, deduce a theoretical analysis of the error introduced in the previous conversion process and based on this theory, realize the conversion of the VGG-16 architecture to SNN on the ImageNet dataset. The accuracy rate of 84.86% is achieved [50]. A comprehensive analysis and comparison of several common algorithms are carried out, and the results are shown in table 2.

B. BASIS OF SC

SC was first proposed in the 1960s to simplify complex binary computing units [51-53]. And SC has attracted much attention due to its fault tolerance and low-cost arithmetic functional units [54], [55], [56]. Because of this characteristic, SC can realize complex operations such as addition, subtraction, and multiplication through simple logic gates [57]. Compared with traditional binary operations, SC can greatly save hardware resource overhead [58].

1) STOCHASTIC MULTIPLIER

The multiplier is the basic calculation unit of SC. Usually, the random sequence is represented by the multiplication

TABLE 2. Performance of SNN algorithm on different datasets

Author/Algorithm	Network Structure	Pulse form	Processing	Dataset	Accuracy(%)
Zhang/SpiKL-IP[44]	CNN-SNN	Real-time learning of input pulse patterns	Online	CityScape Dataset / TI Voice Library	97.78/98.46
Wysoski[46]	4-layer eSNN	Single pulse level time code	Online	Visual Dataset / Sound Dataset	60/40
Dhoble[45]	4-layer dynamic evolutionary SNN	Multi-pulse level time encoding	Online	EGG dataset	83.33
Wang[47]	Multi-layer feed-forward SNN	Population-based sparse coding	Online	Pima diabetes	91.8
Courbariaux[48]	CNN-SNN	Pulse Sequence	Offline	CIFAR-10	91.35
Cao[49]	DNN-SNN	Pulse Sequence	Offline	CIFAR-10	77.43
Rueckauer[50]	VGG-16-SNN	Pulse Sequence	Offline	ImageNet	84.86

operation realized by the AND gate to represent the unipolar type. The multiplication operation realized by the XOR gate is used to represent the bipolar type. x and y respectively represent two mutually independent random sequences, then a single AND gate can be calculated with high accuracy [59]. As shown in figure 5(a), in the unipolar representation of SC, the real number X is interpreted as the probability of a single bit being "1" in a random binary bit stream, i.e., $x = p(X)$. For example, the binary number $x = 0.375$ is interpreted as $p(x) = 3/8$ and can be represented by the bitstream $X = 01001010$.The number of 1s in the bitstream and the bitstream length are 3 and 8 [60]. Note that the unipolar representation is in the range $[0, 1]$, whereas in the bipolar representation of SC, as shown in figure 5(b), the real number x is in the range $[-1, 1]$ and is interpreted as $x = 2P(X) - 1$.

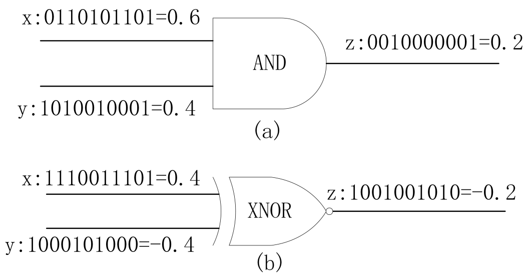


FIGURE 5. SC (a) Unipolar multiplier (b) Bipolar multiplier.

2) STOCHASTIC ADDER

There are many implementations of adders based on SC. Common implementations include three basic structures: OR gate, multiplexer (MUX), and parallel counter (PC). As shown in Figure 6(a), the overhead of the OR gate is the smallest, but when there are multiple SC sequences superimposed, the error will gradually accumulate, reducing the accuracy; as shown in Figure 6(b), the circuit of the MUX structure is a scaled adder. Compared with the OR gate, the accuracy has been improved to a certain extent. However, as the number of inputs increases and the scaling factor increases, the cause is compressed into a small value, causing errors; as shown in Figure 6(c), the PC is a probabilistic adder with high precision, but the hardware overhead and delay are also large. The structure of the approximate unit and parallel addition tree based on alternating APC and OR gate proposed by KIM et al. has certain limitations. The accuracy is high only when the random sequence length is large [61].

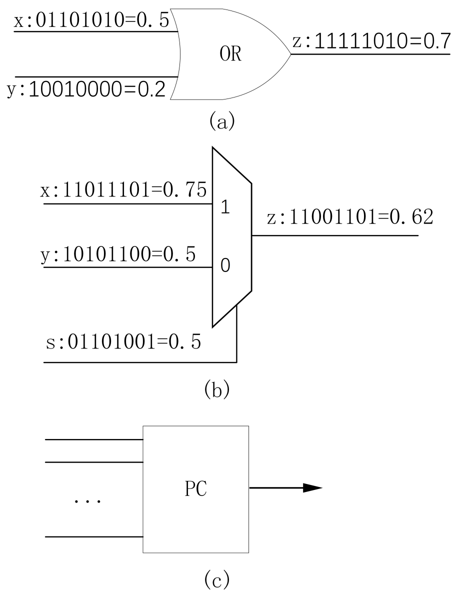


FIGURE 6. Stochastic adder (a) Or-gate based stochastic adder (b) MUX based stochastic b adder (c) Parallel counter.

III. TIME ARCHITECTURE AND COMMUNICATION PROTOCOL

Neural networks are algorithms with inherent parallelism [62]. In addition to the parallel operation of the data when the neural network performs multiplication and addition accumulation (MAC), the training set consists of many samples that can be fed into the network in batches. Use the parallel computing paradigm to exploit the inherent parallelism of layers to improve the performance of hardware implementations of neural networks. In parallel computing solutions, the time and space architectures are different [63]. Both architectures contain processing elements (PEs) that perform parallel operations on the same or other data. In the time architecture, PEs can only access data from central storage, and centralized control, and there is no connection between PEs. Conversely, in a spatial architecture, each PE can also have its control logic and one or more local storage locations. Most importantly, in a spatial architecture, PEs are connected to exchange data with each other, creating a processing array. Figure 7 shows the difference between temporal and spatial architectures.

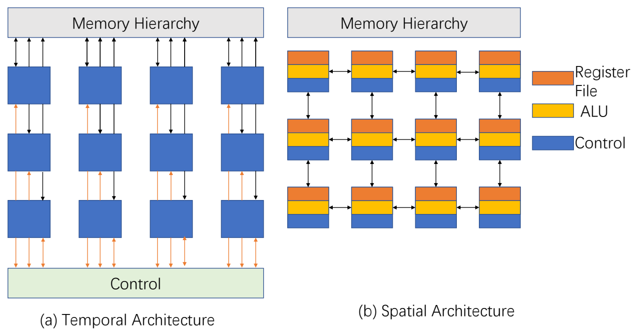


FIGURE 7. Basic models of temporal (a) and spatial (b) architectures.

A. TIME FRAME AND SOFTWARE OPTIMIZATION

Common platforms for time architecture are CPU and GPU. The CPU is a vector processor that can process multiple data elements simultaneously. A vector processor consists of multiple arithmetic logic units (ALUs) that work synchronously and execute instructions on a data vector. Vector processors, on the other hand, use Single Instruction Multiple Data (SIMD) technology.

Among the available hardware platforms, CPUs are often used for SNN inference or training because they offer lower FLOPS and FLOPS/WATT performance. GPUs are architectures with up to thousands of cores designed for parallel computing (e.g., 5120 cores in Nvidia V100 GPU [64]). Similar to vector CPUs, GPUs employ the single instruction multithreading (SIMT) execution model first introduced by Nvidia. The SIMT model executes a single instruction on multiple cores simultaneously. Each core receives different data belonging to multiple threads running in parallel. GPUs are the real workhorse of SNN training and, in some cases inference.

Nvidia GPUs are often used for hardware and software optimization of neural networks. Most neural network frameworks support execution on Nvidia GPUs, such as Pytorch [65], Tensorflow [66], or Caffe [67]. A big advantage of Nvidia GPUs is cuDNN [68], a highly optimized library of DNN primitives. In the latest high-end GPUs, Nvidia combines traditional CUDA cores with tensor cores [64], optimized for large matrix operations. Tensor cores can also support mixed-precision operations. In the new Nvidia A100, the tensor core supports a new format, the tensor format (TF32), which provides a 10 times performance improvement over the performance of the FP32 format on the V100 architecture [69]. In addition, the Nvidia A100's tensor cores can also take advantage of the sparsity of tensors common in DNNs to achieve up to 2 times performance gains.

And a single GPU is composed of multiple stream processors, and each stream processor can process data in parallel. Due to the high parallelism of GPU, the current deep learning widely uses GPU for accelerated training. In essence, the neural network used in deep learning is also inspired by the neural network inside the biological brain. To some extent, the two also have certain similarities, so there are also SNNs developed on the GPU. Schemmel et al. proposed a simulation platform for SNN simulation using GPU [70]. Compared with the CPU, the performance has been greatly improved. At the same time, their proposed simulation platform is also scalable and can simulate large-scale SNNs. Although GPU has unique advantages in parallel computing, there has not been a good solution to its power consumption problem in the face of event-driven spiking neural networks.

B. COMMUNICATION PROTOCOL OF SNN CHIP

When evaluating ANNs, the main performance constraints in terms of throughput and power consumption come from memory bottlenecks [71]-[73]. Inspired by the computing paradigm of the brain, neuromorphic processors aim to allocate memory across an architecture close to the PE . This leads to the parallelization of storage and computation in different layers of the SNN while reducing the power consumption of the entire operation. The computing core is divided into several small neural cores: memory and PE. The layers of the network are distributed among nuclei, each of which stores some of the synaptic weights.

The multi-core parallelism of SNN accelerators relies on a specific network-on-chip (NoC) communication protocol to transmit events among a large number of neural cores with minimal power and latency. In input/output, the core receives/transmits spikes via communication schemes such as address event representation (AER) protocol via (NoC), which encodes event times and organizes connections with low communication costs [74],[75]. As shown in figure 8, such a communication method can greatly reduce the communication bandwidth between chips. This involves sending a packet containing the address of the spiking neuron on a digital bus with asynchronous logic. Once the neuron fires, the address is sent to the NoC, and the firing time is encoded in real-time on the asynchronous bus [76]. This type of architecture can scale as long as routers and control circuits can manage AER requests. Nonetheless, the definition of the number of neurons and synapses per core and the number of cores per chip will limit the class of topologies implemented on a chip. But this limitation can be overcome by implementing a scalable multi-chip architecture [77].

Compared with the frame-driven approach of ANN, the AER protocol has many benefits for large-scale SNN computation [74]. Boahen et al. pointed out that it can reduce

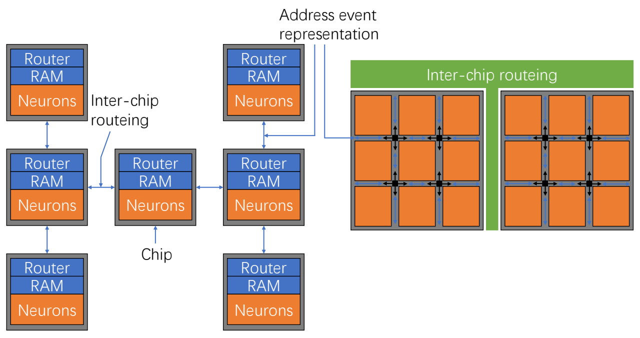


FIGURE 8. Example of AER address time representation.

the size of the network bus while retaining a large connection capacity [75]. Therefore, the AER NoC area requirements are low, enabling large-scale designs. In addition to reducing the transmitted packets to a single address, it also ensures small latency and power overhead. Furthermore, the neuron activity sparsity of the SNN prunes the NoC activity, reducing the number of packets sent on the network.

The NoC communication protocol is easily scalable, and any number of units can be connected as long as the router can manage the requests. Therefore, suitable for multi-chip implementation. However, the power consumption of such a system increases with the number of connected neurons, the average firing frequency of each neuron, and the overall chip size [78], [79]. To address this problem, two different schemes are adopted: one reduces AER's power and latency overhead by exploiting the locality and clustering of neural network algorithms [74], [80]. The other adopts a hierarchical router level, which helps reduce the power impact of large-scale systems [80],[81]. Furthermore, rate and time encoding have significantly different effects on such overhead. Rate encoding usually employs a sequence of Poisson spikes, so a single spike timing error has little effect on the accuracy of the network. Mostafa et al. pointed out that time encoding sends many pulses through the network [82], which can drastically increase the power consumption of the whole system. In temporal coding, timing errors or loss of information can occur when the algorithmic time step duration does not allow the neural core to complete its operation. In this case, pulses may be lost or removed from the network. Hence there is only a trade-off between accuracy and throughput or power [83].

Since the AER protocol is compatible with various NoC routing architectures and broadcast schemes, such as 2D meshes [84]-[86], multicast trees [87], or shrinking rings [88]-[90], therefore, the adopted routing scheme can be adjusted to meet the requirements of the accelerator.

IV. TRADITIONAL SNN CHIP

The neuromorphic computing system imitates the neuromorphic device of biological neurons as the basic unit. The main body is an SNN similar to the neural network approximation in the human brain. Unlike the traditional way of working by following computer instructions, neuromorphic computing systems follow parallel work and distributed processing mechanisms to complete cognitive tasks such as learning, memory, and reasoning [91]. There have been many studies at home and abroad on developing of neural network chips [92]-[94]. At present, the research of neuromorphic chips is mainly divided into three directions: (1) digital-analog hybrid neuromorphic chips designed by analog CMOS circuits for neural computing units, synaptic circuits, and digital CMOS circuits for routers; (2) by all-digital CMOS circuits Design a pure digital neuromorphic chip; (3) design a new type of neuromorphic chip with a new type of resistive memory to design the synapse circuit and part of the computing unit, and a CMOS circuit to design the routing circuit. Three different chips will be described below.

A. DIGITAL-ANALOG HYBRID NEUROMORPHIC CHIP

Neuromorphic engineering based on analog circuits proposed by Mead et al. [95]. Using large-scale digital-analog hybrid circuits to simulate the electrophysiological behavior of real neurons and synapses is more efficient and energy-efficient than using semiconductor devices than traditional CPUs [96]. The most famous is the neuromorphic supercomputer Neurogrid system developed by Stanford University in the United States, which flexibly utilizes the similarities between the dynamic characteristics of neuron ion channels and the electrical characteristics of transistor subthreshold regions to design neuron circuits and synaptic circuits. This is a neuromorphic chip based on an AMS circuit. The chip consists of software, driver, and hardware system. The hardware system consists of a PCB of 16 Neurogrid chips, and the chips are linked through a tree structure [97]-[99]. Each chip contains a square matrix of 256×256 neurons. The resulting single board can simulate large-scale intracerebral neurons and synaptic connections, which can be applied to brain-computer interfaces [100].

The ROLLS chip of the University of Zurich in Switzerland [101], which has only 256 neurons and 128k synapses, is used to simulate the physical activities of the biological nervous system, study computational neuroscience models, and build brain-like computing systems. Unlike traditional Von Neumann processors, the ROLLS neuromorphic processor uses memory and computation co-located. The architecture includes a configurable array of synaptic circuits where spiking neurons produce biologically realistic response properties that express a wide range of real behaviors [102]. The ROLLS chip features online learning algorithms, such as spike-driven synaptic plasticity rules (STDP), that can validate multiple neuromorphic computing modalities. The HICANN chip of Heidelberg University uses a special interconnection technology to interconnect 352 chips in the entire wafer to realize a wafer-level neuromorphic system [103], [104], which is 10,000 times faster than real-time and is used for large-scale parallelism. The HICANN chip is designed to provide scientists with supercomputers to speed up large-scale SNN simulations. The cxQuad chip is a novel multicore device that includes analog neuron and synaptic circuits and an asynchronous digital routing fabric optimized to minimize memory requirements, and maximize scalability and reconfigurability [105].

Hybrid implementations are powerful tools for real-time simulation of large-scale neural networks. Calculations can be performed using analog circuits while maintaining the flexibility of digital programmable devices. The hybrid implementation combines space-saving, power-saving analog circuitry with a scalable binary-digital system, and clockless communication through sparse pulse coding improves computational efficiency. The generated precise time-to-time correspondence improves the time-sensitive performance of neural network scale operations. In hybrid-implemented neuromorphic systems, digital systems implement network connections, while analog electrical circuits are dedicated to reproducing neuronal dynamics. Because the on-chip topology of an analog circuit is usually embedded and fixed after manufacturing, the entire system needs to be prototyped and optimized in an FPGA before manufacturing the analog circuit. Hybrid digital-analog technology represents an advance in neuromorphic systems, integrating large-scale, biorealistic configurable neural networks on a single chip.

B. PURE DIGITAL NEUROMORPHIC CHIP

The advantage of simulating neuromorphic systems is to study the interaction problem with the system and the environment in real-time. Analog neuron circuits only provide a precise deterministic computation of digital analog neurons, which is not ideal for detailed quantitative studies. At the beginning of this century, many purely digital neuromorphic systems appeared. Based on the stability and reliability of digital circuits, these systems can realize large-scale neuromorphic systems with ultra-low power consumption and accurately reproduce algorithms. Many chips can already, to a certain extent, solve real-life problems.

In 2014, IBM Corporation of the United States launched the famous TrueNorth chip [106], which adopted a fully customized ASIC solution. Implementing a specific network model, LIF neuron model, and connection methods that support limited programming and using Samsung's 28-nm technology, the chip consists of 5.4 billion transistors and occupies only $4.3cm^3$ area. Through the asynchronous circuit design, it has extremely low operating power consumption and can achieve a computing scale of 1 million neurons and 256 million synapses. TrueNorth has good scalability, does not rely on the global clock to coordinate work, and does not affect the overall work due to a chip failure. The chip supports deep feedback algorithms so that it can be practically applied in fields such as image recognition and speech processing. The vision application system 161, composed of 3 million neurons implemented by the TrueNorth chip, consumes about 200 mW [107]. The DARPA SyNAPSE system consists of 48 TrueNorth chips interconnected in an array [108], which can realize a neural network with a scale of 50 million neurons. The peak performance of fixed-point computing can reach 266 GB/s. Although IBM uses SRAM, SRAM is a special 8T structure, and it is only possible to achieve the expected scale and power consumption requirements under specific circumstances.

Although the TrueNorth chip, as a breakthrough development in the field of brain-like computing, has greatly promoted the development of artificial intelligence, the TrueNorth chip only supports the reasoning of the SNN. It does not support neuromorphic plasticity, and parameter updates can only be learned on software but not on a chip, so there is still a lot of room for the development of brain-like computing.

SpiNNaker network architecture was developed at the University of Manchester, UK. The exponential function of SpiNNaker is implemented through a lookup table. However, as more complex neural models are developed, the memory requirements grow. To save the limited memory resources in the SpiNNaker chip, a single hardware accelerator for exponential and natural logarithms is specially built and designed through a fixed-point method, which improves energy efficiency with a certain loss of precision. This fixed-point approach enables custom configurations for other systems with different power, area, accuracy, and delay constraints [109],[110]. Part of the brain function model is implemented based on an ARM chip, which can imitate the function of brain regions, and its communication mechanism is suitable for real-time modelling [111].

The neuromorphic chip Loihi released by Intel Corporation combines the STDP model through the pulse transmission data between neurons. The difference from other chips is that the chip has an autonomous learning function [112]. Loihi chips can not only support neuromorphic computing with extremely low power consumption [113], but also support a variety of neuromorphic plasticity, integrating various achievable learning rules, complex neuron models, and various information encoding protocols. Together, multiple algorithms can be simulated. Loihi's chip uses Intel's 14-nm FinFET process and contains 2.07B transistors, which was Intel's fifth-largest chip at the time. With 128 cores integrated into a single Loihi chip and 1024 neural pulse units integrated into each core, SNNs can be run and trained with extremely low power consumption using the Loihi chip. Loihi is the first system that can simultaneously support mechanisms such as sparse network compression, inter-kernel multicasting, variable synaptic formats, and Population-based hierarchical interconnection [114]. Intel also released Pohoiki Springs, a neuromorphic system that can support 100 million neuron computing. The system is built through a data center rack composed of 768 Loihi brain-like computing chips. The overall neural capacity is comparable to that of small mammals. It is the largest brain-like computing system implemented by Intel [115]-[117].

The current neuromorphic chips mainly follow the principles of neurodynamics to build brain-like neural networks, but they are not compatible with mature models such as ANN. In this regard, Tsinghua University has developed an artificial intelligence computing chip called "Tianji", with a size of 3.8*3.8mm2 and a 28nm process. Although the operating frequency is only 300MHz, its performance can reach 1278 GOPs/W. The chip consists of more than 150 computing units, which can satisfy the calculation of about 40,000 neurons and 10 million synapses. The biggest feature of "Tianji" is integrating two different artificial intelligence research directions based on computing science and neuroscience into one platform. This chip can support both existing machine learning algorithms and brain-like computing algorithms. [118]. "Tianji" combines the two technical routes of neuroscience and computer science and adopts a non-von Neumann paradigm. With hybrid compatibility, multi-core architecture, localized memory, and streamlined data flow, it can support cross-paradigm modeling, maximize parallelism, improve power efficiency, and communicate seamlessly between models [119].

Zhejiang University and Zhijiang Laboratory used 792 Darwin 2nd-generation chips to jointly develop Darwin-Mouse, a brain-like computer whose computing scale is equivalent to that of a small mammalian brain [120]. The computer was the largest brain-like computing system in the world at that time and could achieve a computing scale of 120 million neurons and nearly 100 billion synapses. The chip adopts standard 180nm CMOS technology and helps realize applications such as collaborative robot work and EEG signal potential decoding. The overall computing power consumption is between 350-500W. The computer consists of nearly 800 "Darwin 2" chips, each containing 576 computing cores, each of which can realize the computation of about 256 neurons and 10 million synapses [121]. BrainScales is the realization of several interconnected chips, each composed of several HICNN neural cores, to accelerate the time simulation of brain-like neural networks with accurate biological neural behavior [122].

A purely digital implementation consumes more silicon area and power per function but has significantly reduced development time and is not affected by a power supply, thermal noise, or device mismatch. In addition, high-precision digital computing can realize network communication systems with high dynamic range, higher stability, reliability, and repeatability.

C. MEMRISTOR-BASED NEUROMORPHIC CHIPS

Besides being based on conventional CMOS, memristors with desirable properties have become one of the main device choices for neuromorphic computing [123]. The brain's neurons are connected in three dimensions, allowing very dense and highly parallel networks to be implemented on a minimal scale. Neural networks on silicon, on the other hand, are mainly two-dimensional, so they cannot be integrated to achieve similar densities. Many researchers have proposed the latest parallel implementation: crossbar switch arrays [124]-[128]. This design aims to combine the memory and neuron update parts of the neural core in a single unit, resulting in a speed increase and a reduction in energy consumption, and true non-von Neumann computation [129], [124]. It consists of two metal wires intersecting orthogonally, as shown in Figure 9. The nanoelectronic device mimics the behavior of synapses set at each intersection. One direction represents the output of the presynaptic neuron. The other direction represents the connected postsynaptic neuron. Thus, the operation of an analog crossbar array consists of applying voltages on input lines and reading currents on corresponding output lines. The conductance of each device represents the synaptic weight of the connection, the resulting currents are summed according to Kirchhoff's law, and a dot product operation is implemented [126].

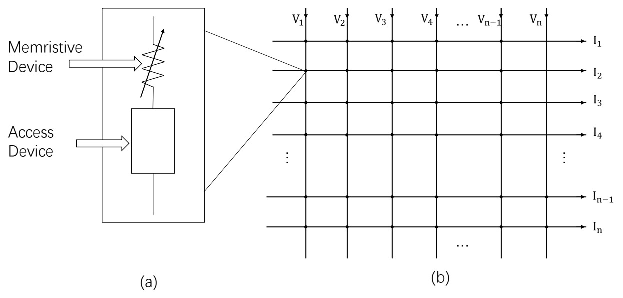


FIGURE 9. Representation of crossbar array implementation. (a) Description of the memristive device implementation. (b) Top view of a typical crossbar design, with the input along the vertical line and the output along the horizontal line.

To achieve high-density crossbar switches without loss of accuracy, the device must be three things: (1) small, (2) low power consumption for reading and write operations, and (3) stable [126]. Pulse code modulationPCM [130], [131] and metal oxide resistive devices [132]-[134] are good candidates for (1) and (2) because their power dissipation decreases with their size. However, on a small scale, they cannot yet meet the third requirement, which greatly limits the use of crossbar switch arrays as accelerators for neural network algorithms. The realization of crossbar switch arrays is still under investigation due to this limitation. Ankit et al. achieved the simulation of SNNs at the placement level of fully neuromorphic architectures [124], achieving huge gains in energy and speed when using crossbar switch arrays compared to traditional neuromorphic architectures. While hundreds of gains are achieved when simulating FC networks and a few tenths when evaluating CNNs, confirming that the crossbar gain is highly dependent on the network topology. Recently, Ambrogio et al. showed that relative to ideal simulation [129], equivalent accuracy was achieved on ANN evaluation using a PCM crossbar switch array controlled externally by a computer through the detector. And the potential power efficiency is one to two orders of magnitude better than a standard CPU or GPU.

One advantage of SNNs over ANNs is that the activations are binary. This also simplifies the surrounding circuitry since the same voltage is applied to each crossbar input. The update of neurons can be implemented in analog circuits or digital circuits. Analog implementations can achieve very high throughput, ideally with a capacitor at the output of each line [135], especially when the memristive element is placed directly between the two access lines. However, due to the lack of control over the intersection design [136], crossbars are usually implemented using access devices, which greatly reduces the design density. A digital implementation of the output circuit would require an analog to digital (A/D)converter and memory to store neuron states [137, 138]. Neuron updates can again be time multiplexed to reduce hardware requirements. In this case, the crossbar array still has an advantage as the computation happens in memory and requires transfer costs to and from neurons' state memory and A/D convert. However, ANNs with full-precision activations on such designs require simplified network topology [139] or improved network circuitry to provide accurate voltage values to each input line [140] and apply nonlinearity after MAC operation.

In conclusion, crossbar arrays with NonVolatile Memory NVM devices are promising in terms of performance and scalability, especially in the case of a fully analog implementation, where neuron updates are implemented in parallel. It has shown good results but still needs improvement to guarantee reliable, fully on-chip, and long-lasting lifetimes.

At present, memristor-based synapses have mimicked several synaptic functions of biological synapses. Most of the currently studied neural networks are mainly based on CMOS circuits, which require many active components to realize the function of neurons. But CMOS-based neurons naturally take up much space and suffer from high power consumption. Therefore, the memristor-based neural network is constructed according to the principles of ANN and SNN. Diffusion memristors can provide a highly desirable kinetic description of synapses and neural functions in neural networks. Synapses are represented in Figure 10(a) by arrays of synapses, connected by axon terminals to the dendrites of individual neurons, similar to the biological scenario in Figure 10(b).

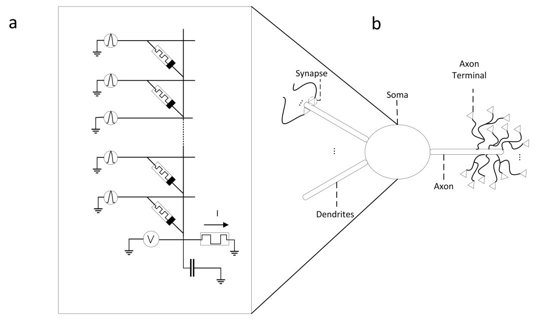


FIGURE 10. (a)synaptic apparatus connected by axon terminals to the dendrites of individual neurons (b)biological scenario.

There are four main types of memristors: redox reaction memristors [141], phase transition mechanism memristors, ferroelectric tunnel junction effect memristors, and magnetoresistive effect memristors. Due to the small surface area and ease of integration, memristors are often used in arrays, such as 3D memristor arrays based on stacking technology and crossbar-type 2D arrays [142]. In addition to high integration, memristor arrays have multiplication and computing capabilities. By applying a voltage to the WL line of the array and reading the current collection on the SL line, the power of multiply-add operations, which have always been the most resource-intensive parts of neuromorphic computing, can be efficiently calculated. Some small networks have demonstrated efficient operations based on memristor arrays [143–145]. In addition to their advantages in computing and integration, memristors are considered devices with synaptic plasticity [146-147]. In addition to synaptic realization, memristors with threshold switching properties are considered to enable the realization of high-density neurons [148]. In addition, memristor neurons can be used in perceptual systems to convert analog perceptual signals into pulsed signals [149].

Memristor-based neuromorphic computing is still in the potential of using device principles to explore neural computing or to verify small networks by constructing small circuit systems, limited by the integration difficulty of the memristor system itself and the limitations of anti-biotic synaptic learning rules. Sexual, large-scale memristor spiking neural networks have remained largely unreported.

D. PROBLEMS WITH TRADITIONAL SNNS

Traditional von Neumann computing architectures suffer from scalability limitations regarding computational speed and power consumption. Novel brain-inspired architectures have emerged as alternative computing platforms, especially for cognitive tasks requiring massive parallel data processing. As discussed in Section 3 above, one of the main bottlenecks in the CMOS implementation of these neuromorphic parallel architectures is the physical implementation of large-scale synaptic interconnections between neurons and synaptic adaptation. Implementing adaptive synaptic connections in CMOS technology requires using many circuits for analog memory or digital memory blocks, which are expensive in terms of area and energy requirements. In addition, learning rules that update these synaptic memory devices must be implemented. Developing compact adaptive devices that conform to biological learning rules to achieve synaptic connections has stimulated research into alternative nanotechnology to complement CMOS technology. Memristive devices are novel two-terminal devices capable of changing their conductance depending on the voltage/current applied to their terminals.

The current mainstream SNN chips usually use pure digital circuits to simulate the functions of neural synapses and neurons to build neuromorphic cores. Connect multiple neuromorphic cores through on-chip routing to form neuromorphic chips and use digital-analog hybrid circuits to simulate neural synapses. The neuromorphic core is constructed with the dynamic changes of neurons, and multiple neuromorphic cores are connected through on-chip routing to form a neuromorphic chip. Although these two technical routes use advanced technology to simulate a neural network with a scale of hundreds of millions of neurons, due to CMOS Circuits are limited by two-dimensional connections and a limited number of interconnected metals and routing protocols, and there are still enormous difficulties in realizing biological brain simulations with 3D structures. One of the major bottlenecks in the CMOS implementation of these neuromorphic parallel architectures is the physical implementation of large-scale synaptic interconnections between neurons and synaptic adaptation. An ideal hardware deep learning system should be able to have online learning capabilities and reconfigurability for different applications. The challenge of designing a highly scalable and parallel hardware deep learning system and providing online learning capabilities needs to span hardware, algorithms, and applications. New computing paradigm. Since the measurement units of parameters such as Synapses, neuron size, number of synapses, number of chip cores, and power consumption are not uniform, there is no way to compare, so only the chip area as shown in figure 11(a) and manufacturing process as shown in figure 11(b) of different SNN models are compared. Summarizes the current mainstream spiking neural network chips and compares their relevant characteristics in table 3.

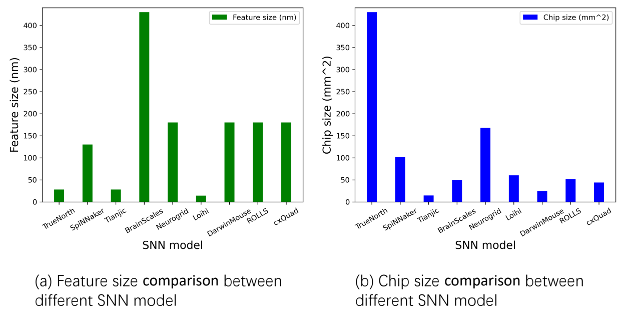


FIGURE 11. compares die area and fabrication process of different SNN models.

V. THE SNN CHIP BASED ON PROBABILITY CALCULATION

SC is considered the next frontier of energy-efficient edge computing [150] because of its energy-efficient operation and ability to tolerate fault tolerance in areas such as recognition, vision, data mining, etc. At the same time, many applications are trying to move challenging workloads from cloud computing to edge devices. Therefore, SC has become a research hotspot.

A. APPLICATION OF PROBABILISTIC COMPUTING IN TRADITIONAL NEURAL NETWORK CHIPS

Deep learning has an increasing demand for energy-efficient, high-computing power, and low-power hardware processing systems. However, computing systems using classical computing architectures encounter the famous "von Neumann bottleneck", "memory wall", and "functionality". Problems such as “wall consumption” severely limit the improvement of the processing energy efficiency of deep neural networks [151]. The computational method of the biological brain is completely different from the von Neumann computing system. Biological neurons use pulse sequences based on time and space encoding to transmit information rather than encoded binary data. The SNN is a neural network that simulates the biological brain. It is completely different from the traditional neural network and requires fewer computing resources. Therefore, studying the neural network deep learning architecture based on impulse power is a breakthrough in solving the computational bottleneck and has new research value.

SC is one of the important realization methods of a neural network [152-154]. In 2001, Brown and Card first applied SC to neural network calculation [155], replacing traditional binary number calculation units with SC units. The calculation results show that the accuracy of the SC unit will decrease, and there are obvious advantages in hardware circuit area reduction, power consumption reduction, and calculation speed improvement. Ardakani et al. proposed an effective scheme for implementing DBN using integral probability. The experimental results show that the system’s delay is reduced by 84%, the hardware occupied area of the modified scheme is reduced by 66%, and the power consumption is reduced by 33%, which effectively improves the calculation efficiency and accuracy [156]. LI et al. proposed an efficient stochastic computing-based large-scale deep convolutional neural network (DCNN) framework, using approximately parallel counters and optimizing the train multiplier, and proposed the stochastic computing-based ReLu activation function for the first time. The results show that the hardware circuit can accurately simulate the function output when the input range of the function is limited to [-5,5] [157]. LI et al. introduced two important technologies, Normalization, and Dropout, in the deep convolutional neural network (DCNN) based on SC and implemented the corresponding functions on the hardware. When using the AlexNet model to verify the ImageNet dataset. The results show that the accuracy of Top-1 is improved by 3.26%, and the accuracy of Top-5 is improved by 3.05% [158].

Ren first proposed the comprehensive design and optimization framework of a deep CNN based on probabilistic computation. Achieves extremely low hardware occupancy, low power consumption, and power consumption while maintaining high network accuracy. Comparing the improved model with the traditional model, SC increases the throughput of the hardware circuit by as much as 100 times [159]. Zhang et al. designed a motor controller based

TABLE 3. Comparison of spiking neural network morphological chips

SNN model	Design	Time	Weight Storage	Neuron size (pcs)	Synapse size(pcs)	Core per Chip	Chip Area (mm^2)	Technology (nm)	Power or energy	Bio-Mimicry	Advantages	Disadvantages
TrueNorth[106]	Digital	Discrete	1b	1 million	256 million	4096	430	28	70mW	LIF	Highly configurable	Only off-camera study is possible
SpiNNaker[111]	Digital	Discrete	Indeterminate	1 billion	100 billion	16	102	130	4K	LIF	Study on film	Excessive volume
Tianjic[118]	Simulation	Discrete	Indeterminate	40,000	10 million	156	14.44	28	1.54pJ	LIF	Compatible with various neural models and algorithms	NA
BrainScales[122]	Simulation	Discrete	4b	196608	50331648	352	50	430	174pJ	AdExp-IF	Highly configurable	Cannot handle actual tasks
Neurogrid[100]	Mixed	Real Time	off-chip	1 million	6 billion	1	168	180	3.1W	LIF	Fast operation speed and low power consumption	Does not reflect synaptic plasticity
Loihi[112]	Digital	Discrete	1 9b	131040	130 million	128	60	14	36mW	LIF	On-chip learning, highly configurable	Cannot handle actual tasks
DarwinMouse[120]	Digital	Discrete	1 15b	2048	4194304	576	25	180	0.84mW	LIF	Highly configurable	Small chip neuron size
ROLLS[101]	Mixed	Real Time	Capacitor	256	128k	1	51.44	180	4mW	LIF	Multiple synaptic plasticity rules and network structures	Smaller
cxQuad[105]	Mixed	Real Time	12b	1k	64k	4	43.8	180	945uW	LIF	Highly configurable	Smaller

on a neural network and implemented its specific functions on FPGA. The neural network is implemented by stochastic computing. Compared with the traditional microcontroller or DSP controller implementation, the experimental results The motor controller was shown to achieve lower cost and higher performance [160].

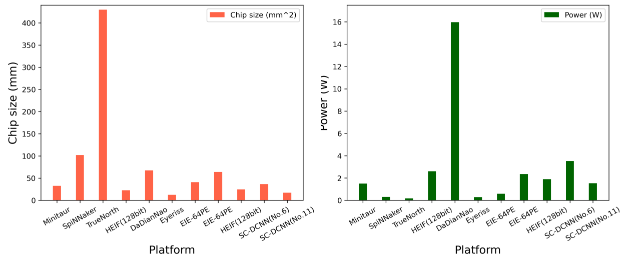
Hirtzlin and Penkovsky et al. proposed to apply SC to binary neural networks, and tests on the Fashion-MNIST and CIFAR-10 datasets showed only a 1.4% drop in accuracy. Still, the circuit area could be reduced by 62% [161]. Sim et al. proposed a matrix-vector probabilistic multiplier [162]. The results show that the multiplier can balance the random sequence length and calculation accuracy and reduce the calculation delay compared with the traditional algorithm. Energy consumption also has a significant effect. Hojabr and Kamyar et al. proposed the SkippyNN architecture [163], a stochastic computation-based convolutional neural network architecture for embedded devices, which can reduce the computation time based on SC in the CNN convolutional layers. Experimental The results show that the SkippyNN architecture achieves 1.2 times the computational speed improvement and 2.7 times the energy saving compared to the traditional binary implementation.

Hojabr and Kamyar et al. proposed the SkippyNN architecture [102], a Stochastic computation-based convolutional neural network architecture for embedded devices, reducing the computation time of Stochastic-based computation in the CNN convolutional layer compared with traditional algorithms. Experimental results show that the SkippyNN architecture achieves a 1.2-fold increase in computational speed and a 2.7-fold energy saving compared to the conventional binary implementation. Xiong et al. use a non-correlation independent non-random encoding of random sequences and apply this encoding to a random multiplier to adjust the sequence length by an adaptive algorithm. The results of the study show that the sequence length was significantly reduced to 64 bits, reducing the overall computational latency [103]. Wang and Zhang et al. designed a non-scaling high-precision random adder applied to a CNN in combination with the Winograd algorithm. The results showed computational accuracy of the arbitrary computation was guaranteed while reducing the hardware complexity of the convolutional operation [104].

Xiong et al. proposed a non-correlated and independent non-random encoding method for random sequences [164]. They applied this encoding method to random multipliers to adjust the sequence length through an adaptive algorithm. The results show that the sequence length is affected by significantly reduced to 64 bits, reducing overall computational latency. Wang and Zhang et al. designed a non-scaling high-precision random adder and applied it to a CNN . Combined with the Winograd algorithm, the results show that the hardware complexity of the convolution operation is reduced while ensuring random computing. The computational accuracy of [165]. Neil et al. propose Minitaur, an event-driven FPGA-based SNN accelerator for investigating the capabilities of FPGA platforms to implement a real-time, event-driven deep spiking network that achieves 92% accuracy on the MNIST dataset. accuracy [166]. Stromatias et al. implemented a spike-based DBN on the SpiNNaker platform, achieving 95% classification accuracy on the MNIST dataset, which is only 0.06% lower than the software implementation, while consuming 0.3 W, the average classification latency is 20ms [167].

For the first time, Esser et al. used the offline training method of backpropagation to create a network that reconciled the incompatibility between the backpropagation algorithm and neuromorphic hardware. The proposed SNN architecture achieved a recognition accuracy of 99.42% on the MNIST dataset and ran the network in real-time on TrueNorth chips [168]. Luo et al. proposed a network consisting of interconnected nodes, each containing logical computation, enhanced dynamic random access memory (eDRAM), and a router structure. This architecture can achieve 450.65 times speedup and 150.31 times cooling reduction on GPU [169]. Chen et al. proposed a CNN accelerator named Eyeriss, which can support the computation of high-throughput CNNs and is optimized for the energy efficiency of the whole system, including the accelerator chip and off-chip dynamic random access memory (DRAM) [170]. Han et al. proposed an Energy Efficient Inference Engine (EIE), which utilized weight sharing and distributed storage computing to accelerate neural networks, showing obvious advantages in energy consumption and hardware area reduction [171]. Ren et al. adopted a top-down approach to design an optimization framework for SC-based DNNs, fully using SC's advantages, significantly reducing the hardware area and achieving low power consumption while maintaining high network accuracy [172].

Table 4 shows the parameter comparison of neural networks based on SC and other software and hardware platforms. It can be seen that there is a theoretical and time basis for applying SC to neural networks. The application of neural networks on the device side is greatly limited, mainly because embedded devices cannot provide enough computing power, storage units, and bandwidth. The bit-wise execution of SC can greatly reduce the hardware complexity. On the other hand, the neural network model usually has multiple passes and iterations in the inference process, which also makes the random error in the inference process less likely to occur. It severely impacts the overall accuracy, which also matches the highly error-tolerant nature of stochastic computing. Compared with traditional binary computing, SC significantly improves computing speed and computing energy efficiency and reduces resource consumption. Since the parameters such as accuracy, Throughput, Area Efficiency, and Energy Efficiency are too different, only the chip area as shown in figure 12 (a) and power consumption as shown in figure 12(b) are compared.



(a) Chip size comparison between different platform

(b) Power comparison between different platform

FIGURE 12. compares die area and power consumption of different platforms.

Compares the stochastic computing-based neural networks and other software and hardware platforms in table 4 . It can be seen that there is a theoretical and temporal basis for applying SC to neural networks. The bit-by-bit execution of SC can massively reduce the hardware complexity. On the other hand, there are usually multiple passes and iterations in the inference process of neural network models, which makes the random errors in the inference process do not have a serious impact on the overall accuracy, which also matches the highly error-tolerant nature of Stochastic computation. Compared with traditional binary calculation, SC is significantly faster, more energy-efficient, and less resource-consuming.

B. APPLICATION OF SC IN SNN CHIP

The SNN is a new data storage and computing technology based on neural networks. Simulating the working mechanism of the brain can break through the von Neumann bottleneck encountered by traditional computers when dealing with large-scale problems and significantly improve the speed of information processing. Significantly reduces power consumption and has self-learning and adaptive capabilities.

In recent years, some researchers have attempted to reduce power consumption and area overhead while retaining the original advantages of SNN. Kuang and Wang et al. proposed to use the Euler approximation method to design the LIF neuron model to solve the problem of non-differentiable impulses [173]. Using the Euler method to simplify and accumulate neuron models can significantly reduce the computational complexity of SNNs. However, this approach still suffers from a large area overhead due to the many multipliers involved.

As a unique data representation and processing technology, stochastic computing many complex arithmetic operations can be implemented using simple logic gates in the SC framework, providing a huge design space for neuron integration. And SC has strong fault tolerance, because in SC, data is processed in the form of a bit stream, and these data are interpreted as probabilities. So SC enables fully parallel and scalable hardware implementation of large-scale deep learning systems.

TABLE 4. Comparison between the neural network with the addition of SC method and other neural networks

Neural Network Model	Network type	Dataset	Platform	Area (mm^2)	Power (W)	Accuracy(%)	Throughput (Images/s)	Area Efficiency (Images/s/mm2)	Energy Efficiency (Images/J)
Minitaur[166]	ANN	MNIST	FPGA	32.5	1.5	92.00	4880	N/A	$\geq$ 3253
SpiNNaker[167]	DBN	MNIST	ARM	102	0.3	95.00	50	N/A	166.7
TrueNorth[168]	SNN	MNIST	ASIC	430	0.18	99.42	1000	2.3	9259
HEIF(128bit)[157]	CNN	MNIST	ASIC	22.9	2.6	99.07	3203125	139874	1231971
DaDianNao[169]	CNN	ImageNet	ASIC	67.7	15.97	N/A	147983	2185	9263
Eyeriss[170]	CNN layer	ImageNet	ASIC	12.25	0.28	N/A	35	2.8	125
EIE-64PE[171]	AlexNet	ImageNet	ASIC	40.8	0.59	N/A	81967	2009	138927
HEIF(128bit)[157]	AlexNet	ImageNet	ASIC	24.7	1.9	N/A	2520161	102030	1326400
SC-DCNN(No.6)[172]	CNN	MNIST	ASIC	36.4	3.53	98.26	781250	21439	221287
SC-DCNN(No.11)[172]	CNN	MNIST	ASIC	17.0	1.53	96.64	781250	45946	510734

The rate coding method in SNN is similar to the SC coding in SC-based computation [174]. However, the results of the SC method depend on the correlation of the input pulse sequences involved. When two (or more) pulse trains are used as the input to the SC circuit, the cross-correlation between them will affect the computational accuracy [175]. If the pulse train involved has high cross-correlation, then the pulse train output through the SC circuit has low randomness, and vice versa. On the other hand, many researchers use finite state machine (FSM) processes to implement many nonlinear functions to improve accuracy [176]. By following this design concept, Smithson et al. proposed using the FSM process to implement the LIF neuron model and its hardware architecture [177], [178]. However, this approach still suffers from a large area and power overhead.

To reduce the area overhead, Chen and Kou et al. proposed to use SC adders and multipliers to implement low-cost SNN neurons [179]. Using a large-scale SNN structure, to further reduce the calculation error caused by the cross-correlation of the bit stream. The pruning method can be used to avoid unnecessary calculations in the SNN process, which makes the pulse transmission between each SNN neuron layer. Becoming sparse also helps to improve computational accuracy. Using 40nm process technology to implement the SC-based SNN architecture and analyze the hardware efficiency can save 72.38% to 75.64% of the area overhead and 81.37% to 90.58% of the power consumption compared to the SNN model without the SC method. Xiao et al. proposed an adaptive exponential integral-excited neuron model SC-AdEx based on SC, using probability integrator and AND gate as the basic calculation unit. Compared with AdEx without SC, it occupies a larger area small, faster, and has a lower cost [180].

A high-precision SC-SNN hardware design framework proposed by Tang and Han utilizes the cumulative distribution function of the input signal to generate pulse trains and a priority encoder to convert these pulse trains into index-based signals. In this way, the connection between neuron layers is reduced from $O(N^2)$ to $O(NlogN)$, which solves the problem of relatively low information density and realizes efficient hardware design. Implemented on FPGA for classifying the MNIST dataset, experimental results show that almost the same accuracy as ANN is achieved [181]. Liu and Liang et al. proposed an efficient hardware tripartite synapse structure based on SC. SC is used to replace conventional computing components such as DSP in hardware devices, and the extended SC logic is used to scale the data range during the computing process. The results show that the proposed hardware architecture has the same output as software simulation with lower hardware resource consumption so that it can be applied to large-scale SNNs [182]. Chen and Song et al. designed a probabilistic spiking neuron and realized the reconfigurable computing architecture of the neural network. 8.82 times that of a conventional binary accumulator [183]. Gao and Chen et al. proposed an asynchronous architecture of SNN based on SC, which realized the forward inference operation of SNN based on LIF neurons with 784 inputs and 10 outputs. Used the method of SC to convert the numerical value into a pulse sequence and realized further reduced the power consumption of circuits and systems [184].

SC has been widely used because it can reduce the energy cost of hardware computing [185]. Two main approximation strategies are used for neural network applications: network compression and classical SC.

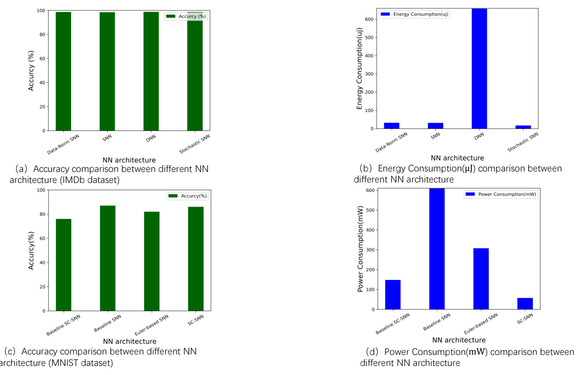
Because neural networks have too many parameters, researchers targeting embedded applications began to reduce weights and activation accuracy to reduce the memory footprint of ANNs, a method known as network compression or quantization. Also, due to the fault tolerance of neural networks and their ability to compensate for approximations while training, the reduced bit precision results in only a small loss of precision [186]-[189]. When implemented in hardware, weight quantization (WQ) shows an energy gain of 1.5 to 2 times with less than a 1% loss in accuracy [190], [191]. Rathi et al. achieved an accuracy loss of about 3% with an energy gain of 2.2 to 3.1 times [192]. There can be a trade-off between the accuracy of the SNN application and the energy and area requirements of the neural network. SC can also implement computational circuits of neurons, where unimportant cells can be deactivated to reduce the computational cost of evaluating SNNs [193].

Training ANNs with random synapses leads to better generalization and has already yielded better accuracy on the test set [194, 195]. The same method applies to SNNs. Spikes with synaptic randomness FPGA implementations of neuromorphic systems have been shown to improve the accuracy of the network while reducing memory requirements [196]. And nanoelectronic devices with inherent cycle-to-cycle variability, such as memristors [197] or VO_2 [198], can reduce the area and power overhead of random number generation. Chen et al. [199] also exploited probabilistic rewiring to increase their throughput, with fewer synapses meaning fewer spike integrations, and thus an increased algorithmic time step. Experimental results show an 8 times speedup and a 7.3 times increase in energy gain, and the accuracy loss in MNIST digit recognition is only 0.25%, from 98.15% to 97.9%. Thus, randomized and quantized synapses can significantly reduce the memory requirements and power consumption of SNN accelerators, and can even be further reduced by pruning insignificant weights. Another approach is to design PEs that approximate their computations by employing modified algorithmic logic units [202]. Jin et al. have shown that when evaluating SNNs on neuromorphic hardware for character recognition [203], carrying-skip adders can achieve 2.4 times and 43% faster speed and energy gains, respectively, with an accuracy loss of only 0.97 %.

Therefore, software and hardware levels SC methods can significantly advance power consumption and speed. However, as the complexity of the dataset increases. with the depth of the network topology, such as the use of ResNet on ImageNet [200], the accuracy loss becomes a non-negligible factor [201].

Shows the classification results on the MNIST dataset based on random SNN, DNN, SNN, and optimized SNN [213] and also compares the energy consumption in table 5. The performance of SNNs on the very simple MNIST image recognition dataset is still marginal; the test accuracy is less than 95% [214] [215], and Smithson et al. found that spiking neurons perform the same as SC when performing rate encoding. The proposed SNN based on SC can further reduce the power consumption of the hardware and achieve 95% accuracy on the MNIST dataset [216].

DNN uses the RELU activation function, and SNN uses the RELU function to convert into IF neurons. Then, from [217] we find the energy consumption required for each operation. The results in Table 5 show that our proposed scheme achieves almost the same performance as the original DNN and better performance than the state-of-the-art SNN. Furthermore, random-based SNNs are more energy efficient compared to other networks. At the same time, Figure 13(a) represents the recognition accuracy of the neural network on the IMDb dataset; Figure 13(b) represents the energy consumption of the neural network on the IMDb dataset; Figure 13(c) represents the recognition accuracy of the neural network on the MNIST dataset; Figure 13(d) shows the power consumption of the neural network on the MNIST dataset..



(a) Accuracy comparison between different NN architecture (IMDb dataset)

(b) Energy Consumption(μJ) comparison between different NN architecture

(c) Accuracy comparison between different NN architecture (MNIST dataset)

(d) Power Consumption(mW) comparison between different NN architecture

FIGURE 13. compares Accuracy, Energy Consumption (J), and Power (mW) for different NN architectures.

C. DELAY PROBLEM OF SC

SC uses discrete pulse sequences to replace sequential binary numbers to achieve lower computing resource consumption. However, its latency or computational accuracy sacrifice also becomes a challenge for hardware design. In response to the above problems, some researchers have made preliminary attempts. Lu et al. proposed a new architecture to implement the fast Winograd algorithm on FPGA [204], reducing the calculation delay and improving the accuracy. Mathieu et al. used FFT and the convolution theorem to reduce the arithmetic complexity of convolutional layers [205], and Vasilache et al. improved a fast Fourier transform convolution implementation based on NVIDIA's cuFFT library [206], and in NVIDIA cuDNN implemented in the library. Strassen's algorithm [207] for fast matrix multiplication was used by Cong et al. [208] to reduce the number of convolutions in a convolutional network layer, thereby reducing its overall arithmetic complexity.

Lavin et al. proposed a new fast algorithm for convolutional neural networks [209], which is based on the minimal filtering algorithm discovered by Toom [210] and Cook [211] and popularized by Winograd [212]. Compared to direct convolution, this algorithm can reduce the arithmetic complexity of convolutional layers by up to 4 times. Arithmetic is performed by dense matrix multiplication of sufficient dimensions. Memory requirements are also low compared to traditional fast fourier transform(FFT) convolution algorithms. These factors make practical implementation possible. And achieved state-of-the-art throughput for all measured batch sizes from 1 to 64 for the NVIDIA Maxwell GPU implementation, found to use up to 16MB of workspace memory simultaneously.

VI. CHALLENGES AND THE ROAD AHEAD

Artificial intelligence and deep learning are already being applied in many different areas, and in the coming years, AI will be the economy's driving force. The development and popularization of artificial intelligence applications are closely related to technological progress. The algorithm is deployed on a chip consisting of several devices implemented

TABLE 5. Comparison of performance accuracy and Power for different NN architectures.

NN architecture	Dataset	Accurcy(%)	Power Consumption(mW) or Energy Consumption(μJ)
Data-Norm SNN [174]	IMDb	98.64	31.8μJ
SNN[174]	IMDb	98.48	31.5μJ
DNN[174]	IMDb	98.68	657.75μJ
Stochastic SNN[174]	IMDb	98.65	17.2μJ
Baseline SC-SNN[216]	MNIST	76	147.99mW
Baseline SNN[214]	MNIST	87	608.56mW
Euler-based SNN[215]	MNIST	82	307.70mW
SC-SNN [179]	MNIST	86	57.33mW

in a certain technology, such as CMOS technology. The growth in the number and complexity of AI applications places increasing performance requirements on hardware (application-driven development). On the other hand, the development of new technologies and hardware improvements allow the development of more complex and, therefore, more accurate applications. The two development directions continue to complement each other and form a virtuous circle. To maintain such a high growth rate, industry and academia will face new challenges in the coming years, proposing two possible development directions.

A. DEVELOPMENT PROSPECTS IN TERMS OF HARDWARE IMPLEMENTATION

The advent of memristors and their synapse-like behavior opened up the possibility of overcoming the limitations of CMOS technology. Memristors can be as small as a few nanometers, but can be densely packed in two-dimensional layers with nanoscale spacing, potentially providing higher neuron and synapse densities. Since the manufacturing process is much cheaper than CMOS, the memristor layers can be stacked in 3D. This approach can achieve the neuronal and synaptic densities of the human brain on a single plate. Furthermore, the tight 3D dense packing between the CMOS neural computing unit and the memristive adaptive memory synaptic element can significantly reduce the current consumption of the final system [129].

The 3D integration technology centered on TSV through-silicon interconnection technology mainly affects the interconnection structure between chips, so this technology mainly reduces the circuit board area required for interconnection between chips. This technology is generally implemented by vertically stacking multiple memory or logic function chips, and connecting the TSVs made in the upper layer of the stacked structure to the bond pads on the top of the lower chips. . However, at this time, each layer of chips in the stacked structure adopts its own design and is still a traditional two-dimensional structure, so the circuit-level interconnection inside each layer of chips is still a traditional two-dimensional design.

In contrast, in the monolithic 3D technology, the 3Dization of the interconnection layer inside the chip is more thorough, so people usually call this technology "true 3D integrated design". At this time, each layer of chips in the chip stack structure is designed as a functional unit in the whole, so that each layer of chips in the stack structure can use the same interconnection structure inside , so this design can further reduce the length of interconnect lines. Moreover, due to the unified design, the area occupied by the signal relay circuit and the like is also smaller, so the overall footprint of the chip can be smaller.

3D integration technology brings high bandwidth advantages; shorter interconnect designs and potentially high parallelism. Circuits can be interconnected on multiple planes and routed vertically through the planes. Using 3D techniques to improve neuromorphic computing efficiency by implementing 3D layers layer by layer [218, 219], it is also possible to separate memory and logic parts on different layers [219, 220]. Zhang et al. proposed that in monolithic 3D [218], the implementation of digital neuromorphic chips for formal processing is more performant if both memory and logic are distributed across multiple chips, which can save relative to the same 2D implementation about 20% power. However, regarding speed, Kim et al. proposed that stacked memory can greatly improve throughput relative to traditional memory-to-side implementations [220]. In addition, some researchers utilize through-silicon via technology processes to limit interconnect density [221, 222] to design analog neuron models with reduced capacitance footprints, increasing throughput and reducing power consumption. However, 3D circuits are not yet a mature technology [223, 224]. 3D technology has no unique constraints in terms of design but also has high process costs. Furthermore, since the AER protocol already allows low-power communication between neural cores, further work is required to understand to what extent 3D techniques can improve the performance of SNN accelerators.

Except for crossbar arrays and 3D circuits. Morro et al. propose to replace part of an ASIC designed with traditional digital gates with neuromorphic hardware [225]. This hybrid neuromorphic integration may be relevant for other applications. Yousefzadeh et al. developed a chip for evaluating formal ANNs that uses event-driven communication between layers [226], a topology they refer to as a hybrid neural network. It requires circuitry to convert asynchronous event-driven information into frames and vice versa. They employ the AER protocol between layers, where the information is encoded in 4 bits, thus guaranteeing minimal AER bus width overhead. For each non-zero activation, a single-word packet is sent through the asynchronous NoC, which is enough to transmit the complete information from one neuron to another. This technique can combine the advantages of SNNs and ANNs while mitigating their disadvantages.

B. DEVELOPMENT PROSPECTS OF SC IN SMART CHIPS

By comparing the application of SC in various neural networks, it can be found that the neural network based on SC has great advantages in the area and power consumption. Due to the extreme parallelization of the SC circuit, all data can technically be preloaded into local memory before the start of the SC cycle. At the same time, a random stream can take hundreds or even thousands of clock cycles to complete (each clock for each random bit). SC can pipeline all SNN arithmetic operations from top to bottom, with all bits at a particular moment in each SC clock cycles through all SNN layers. Therefore, the memory bandwidth bottleneck is not a problem in SC circuits. The arithmetic circuits in SC allow massive parallelization, which benefits SNN hardware implementation in edge computing applications. This advantage is prominent when noise margin is essential at higher clock speeds when parallelizing large SNN models with big data.

Due to the high degree of parallelism of SCs, SC designs can achieve similar performance to traditional binary designs. These advantages make stochastic computation-based SNNs a potential competitive candidate in resource-constrained applications. Despite the strong parallelism of SC, the data bandwidth bottleneck remains a significant challenge. To address this, the algorithm can be modified to reduce the number of data items used (for example, model compression, pruning, or quantization).

One solution is increasing memory bandwidth, which is what high-bandwidth memory (HBM) is for, a stacked DRAM integrated with processing elements through a silicon interposer. The bandwidth of a single HBM2 block is 256GB/s, which is lower than the 616GB/s bandwidth of more traditional Graphics Double Data Rate 6 (GDDR6) memory. However, a stack with four HBM blocks achieves a bandwidth of 1TB/s. HBM2 memory is currently used for Nvidia V100 and P100 GPUs.

Another option is in-memory computing (IMC), which involves moving logic in memory. IMC enhances SNN acceleration by reducing the latency and power consumption required to access memory hierarchies in traditional von Neumann architectures. In addition, parallelization is increased by processing all memory cells simultaneously.

SC takes hundreds or even thousands of clock cycles to complete so that data transfers can be pipelined and buffered asynchronously. Furthermore, a large amount of data needs to be prepared in addition to SC elements. Therefore, limitations of local storage elements such as SRAM (ASIC term) or BRAM/Flip Flop (FPGA term) should be a concern. In any case, memory-centric computing design should be the direction of SC development, especially in SC SNNs, where hundreds of thousands or even millions of operations can be parallelized. Since most modern FPGAs consist of 6-input lookup tables, there is still much room for optimization in implementing SC on FPGAs. ASIC logic may not translate efficiently to FPGA fabric because lookup tables are hardwired. Although FPGAs are flexible in terms of hardware implementation, they are not as customizable as ASICs. Modern FPGAs also include other resources capable of performing performance calculations, such as digital signal processors or arithmetic logic waiting to be used. There are also challenges in overcoming randomness, further improving classification accuracy, and at the same time maintaining high energy efficiency. However, with the application of SC designs in large networks of SNNs, SC provides an alternative, scalable solution for the hardware implementation of spiking neural networks with the potential for efficient machine learning.

VII. SUMMARY

This paper summarizes five neuron models, coding methods, network topology and learning algorithms commonly used in SNNs, and introduces the basic principles and application scenarios of SC. On this basis, three traditional spiking neural network chips, namely digital-analog hybrid neuromorphic chip, pure digital neuromorphic chip and memristor-based neuromorphic chip, are reviewed. The relative characteristics and advantages and disadvantages of synapse scale, chip area and manufacturing process are compared and summarized. It can be seen that SNN chips have opened up a new way for high-performance neural computing platforms to realize low-power neural network computing. However, the research on SNN chips is not mature and is in a stage of rapid development, facing many challenges. At present, there are mainly the following: Several problems: First, the parameters of the processor are highly configurable and have the ability to accurately handle actual tasks; the second is to realize the software-hardware correspondence to run the same program on the simulator and the chip, so that the parameter update can not only be It can be implemented in software, and parameters can be adjusted on the chip. It can support the parallel computing of SNN algorithm and traditional ANN algorithm, making the chip universal. The third is to use extremely low power consumption to run and train SNN to realize low power consumption of neural network. edge computing. It is still difficult to design a low-power, highly scalable and parallel SNN chip.

The advent of memristors opened up the possibility of overcoming the limitations of CMOS technology. Memristors are small in size but densely packed in two-dimensional layers with nanoscale spacing, providing greater neuronal and synaptic density. Memristors are made in a much cheaper process than CMOS and can be stacked in 3D. This approach can achieve the neuronal and synaptic densities of the human brain on a single plate. Furthermore, the tight 3D dense packing between the CMOS computing unit and the memristive adaptive memory synaptic element can significantly reduce the current consumption of the final system. 3D integration technology brings the advantages of high bandwidth, shorter interconnect designs and potentially high parallelism. Circuits can be interconnected on multiple planes and routed vertically through the planes. Leverage 3D technology to improve neuromorphic computing efficiency by implementing 3D layers layer by layer.However, memristor-based neuromorphic computing is still in the early stage of research, and the main research is still to verify the possibility of realizing neural computing with a single device in principle or to conduct small-scale experiments by building a small-scale non-reconfigurable memristor network. Achieving large-scale multi-core reconfigurable memristor neuromorphic chips remains a challenge.

SC is a logic calculation that converts binary numbers into probability-encoded digital pulse code streams, which has the advantages of extremely low area, low power consumption, and high energy efficiency. And the application of probability calculation in traditional neural network chips can not only solve the problems of high power consumption and large memory bandwidth of traditional von Neumann architecture processors but also maintain a high accuracy rate. The SC is added to the SNN chip to realize a computing circuit with extremely low power consumption and high computational efficiency at the milliwatt level. Based on the SC, the traditional ANN and SNN algorithm can be implemented, and ANN transformation can also be completed. For the calculation of SNN, the chip is universal. Although some achievements have been made in building neural network chips using SC, there are still shortcomings. The first is that the accuracy of SC is closely related to the correlation of the input sequence. Second, increasing the sequence length can increase the calculation's accuracy. Still, an excessively long sequence will lead to long delays and low throughput, making it difficult for the network to operate at high speed and in the real-time application below. To improve the computational performance of SC, reducing the sequence length and combining it with the Winograd algorithm can reduce the computational delay and energy consumption. It is also possible to avoid unnecessary computation during the SNN operation by a pruning-based method. It can also alleviate the irrelevance between input signals, making the transmission of spikes between each spiking neuron layer sparse, which also helps Improve calculation accuracy.

The SNN based on SC retains the original advantages of the SNN. It provides a new idea for the chip design based on the SNN. However, there are still several problems. First, it is impossible to use the greatest advantages of neuromorphic chips in terms of efficiency and energy consumption for on-chip learning; second, the high configurability of system parameters and the diversification of configurable parameters have not yet been achieved. However, with the rapid development of artificial intelligence, the development of SNN chips will further tap the potential of ANNs.

REFERENCES

[1] H. Soula, A. Alwan, and G. Beslon, "Learning at the edge of chaos: Temporal coupling of spiking neurons controller for autonomous robotic," in *Proceedings of the AAAI spring symposia on developmental robotics*. AAAI Palo Alto, CA, 2005.

[2] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.

[3] L. Cheng and Y. Liu, "Spiking neural networks: model, learning algorithms and applications," *Kongzhi Yu Juece Control Decis*, vol. 33, no. 5, pp. 923–937, 2018.

[4] Y. Bengio, *Learning Deep Architectures for AI*. Now Publishers Inc., 2009.

[5] Z. Wu, Y. Zhou, Z. Shi, C. Zhang, G. Li, X. Zheng, N. Zheng, and G. Pan, "Cyborg intelligence :recent progress and future directions," 2016.

[6] Zhaohui, Zheng, Xiaoxiang, Tian, Liwen, Kedi, Wang, Yueming, Pan, and Gang, "Visual cue-guided rat cyborg for automatic navigation," *IEEE computational intelligence magazine*, vol. 10, no. 2, pp. 42–52, 2015.

[7] 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*, 2016.

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

[9] D. B. Thomas and W. Luk, "Fpga accelerated simulation of biologically plausible spiking neural networks," in *FCCM 2009, 17th IEEE Symposium on Field Programmable Custom Computing Machines, Napa, California, USA, 5-7 April 2009, Proceedings*, 2009.

[10] M. Singh and S. M. Nowick, "Acm journal on emerging technologies in computing systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 16, no. 1, p. 11, 2010.

[11] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, and D. S. Modha, "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, no. 10, pp. 1537–1557, 2015.

[12] B. D. Brown and H. C. Card, "Stochastic neural computation. i. computational elements," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891–905, 2001.

[13] S. C. Smithson, K. Boga, A. Ardakani, B. H. Meyer, and W. J. Gross, "Stochastic computing can improve upon digital spiking neural networks," in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2016.

[14] S. R. Faraji, M. H. Najafi, B. Li, D. J. Lilja, and K. Bazargan, "Energy-efficient convolutional neural networks with deterministic bit-stream processing," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019.

[15] W. Lu, G. Yan, J. Li, S. Gong, and X. Li, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks," in *2017 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2017.

[16] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi, "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks," in *Design Automation Conference*, 2016, pp. 1–6.

[17] Y. Liu, Y. Wang, F. Lombardi, and H. Jie, "An energy-efficient online-learning stochastic computational deep belief network," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2018.

[18] D. Jenson and M. Riedel, "A deterministic approach to stochastic computation," in *International Conference on Computer-aided Design*, 2016.

[19] Z. Zhang, R. Wang, Z. Zhang, Y. Zhang, and R. Huang, "Circuit reliability comparison between stochastic computing and binary computing," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2020.

[20] M. L. Minsky and S. Papert, "Perceptrons : An introduction to computational geometry," *The MIT Press*, 1991.

[21] D. J. Amit and N. Brunel, "Dynamics of a recurrent network of spiking neurons before and following learning," *Network: Computation in Neural Systems*, 2009.

[22] D. Howard, L. Bull, and B. Costello, "Evolving unipolar memristor spiking neural networks," *Connection Science*, vol. 27, no. 4, pp. 397–416, 2015.

[23] N. Razzali, M. Rahim, D. Daman, and M. Kasmuni, "A design consideration of neuron 3d reconstruction visualization for neuronal morphology," in *IEEE Malaysia International Conference on Communications*, 2010.

[24] A. L. Hodgkin and A. F. Huxley, "Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo." *The Journal of physiology*, vol. 116, no. 4, p. 449, 1952.

[25] A. L. Hodgkin and A. F. Huxle, "A quantitative description of membrane current and its application to conduction and excitation in nerve." *Journal of Physiology*, vol. 117, 1952.

[26] J. Feng, "Is the integrate-and-fire model good enough?–a review." *Neural Networks the Official Journal of the International Neural Network Society*, vol. 14, no. 6-7, pp. 955–975, 2001.

[27] XuYan, YangJing, and ZhongShuiming, "An online supervised learning method based on gradient descent for spiking neurons," *Neural Networks*, 2017.

[28] S. Xiang, Z. Ren, Y. Zhang, Z. Song, and Y. Hao, "Training a multi-layer photonic spiking neural network with modified supervised learning algorithm based on photonic stdp," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. PP, no. 99, pp. 1–1, 2020.

[29] Z. F. Zhang, C. J. Han, and L. H. Liu, "Compartmental model computation and optimization baesd on single thalamocortical neuron," *ence Technology Engineering*, 2009.

[30] Izhikevich and M. E., "Simple model of spiking neurons," *Neural Networks, IEEE Transactions on*, 2003.

[31] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain Research Bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999.

[32] W. Gerstner, "Time structure of the activity in neural network models," *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, vol. 51, no. 1, pp. 738–758, 1995.

[33] Y. C. Wang and H. U. Hua, "New development of artificial cognitive computation:truenorth neuron chip," *Computer Science*, 2016.

[34] Y. H. Liu and X. J. Wang, "Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron," *Journal of Computational Neuroscience*, vol. 10, no. 1, pp. 25–45, 2001.

[35] M. F. Carfora, E. Pirozzi, L. Caputo, and A. Buonocore, "A leaky integrate-and-fire model with adaptation for the generation of a spike train," *Mathematical Biosciences Engineering*, vol. 13, no. 3, pp. 483–493, 2017.

[36] H. A. Le?Thi and J. Judice, "International conference on modelling, computation and optimization in information systems and management sciences," *Computational Optimization Applications*, vol. 50, no. 3, pp. 463–464, 2011.

[37] E. M. Izhikevich, "Dynamical systems in neuroscience : the geometry of excitability and bursting," *MIT Press,*, 2006.

[38] T. Huang, Z. Yu, and Y. Liu, "Brain-like machine: Thought and architecture," *Journal of Computer Research and Development*, 2019.

[39] R. Vanrullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends in Neurosciences*, vol. 28, no. 1, pp. 1–4, 2005.

[40] 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, no. 99, pp. 1–1, 2015.

[41] Jacques, Gautrais, , , Simon, and Thorpe, "Rate coding versus temporal order coding: a theoretical approach," *Biosystems*, vol. 48, no. 1-3, pp. 57–65, 1998.

[42] S. Yin, S. K. Venkataramanaiah, G. K. Chen, R. Krishnamurthy, and J. S. Seo, "Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations," in *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2017.

[43] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition," *Neural Networks*, vol. 41, no. 5, pp. 188–201, 2013.

[44] P. Joshi and J. Triesch, "Rules for information maximization in spiking neurons using intrinsic plasticity," in *International Joint Conference on Neural Networks*, 2009.

[45] K. Dhoble, N. Nuntalid, G. Indiveri, and N. Kasabov, "Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012.

[46] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Fast and adaptive network of spiking neurons for multi-view visual pattern recognition," *Neurocomputing*, vol. 71, no. 13-15, pp. 2563–2575, 2008.

[47] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Adaptive spiking neural networks for audiovisual pattern recognition," *Springer Berlin Heidelberg*, 2008.

[48] J. Wang, A. Belatreche, L. Maguire, and T. M. Mcginnity, "An online supervised learning method for spiking neural networks with adaptive structure," *Neurocomputing*, vol. 144, no. nov.20, pp. 526–536, 2014.

[49] M. Courbariaux, Y. Bengio, and J. P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *International Conference on Neural Information Processing Systems*, 2015.

[50] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

[51] R. Bodo, L. Iulia-Alexandra, H. Yuhuang, P. Michael, and L. Shih-Chii, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017.

[52] Armin, Alaghi, John, P., and Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, 2013.

[53] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[54] J. P. Hayes, "Introduction to stochastic computing and its challenges," in *Design Automation Conference*, 2015.

[55] "Skippynn: An embedded stochastic-computing accelerator for convolutional neural networks," in *ACM/IEEE Design Automation Conference*, 2019.

[56] G. Maor, X. Zeng, Z. Wang, and Y. Hu, "An fpga implementation of stochastic computing-based lstm," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, 2019.

[57] M. Lunglmayr, D. Wiesinger, and W. Haselmayr, "Design and analysis of efficient maximum/minimum circuits for stochastic computing," *IEEE Transactions on Computers*, vol. 69, no. 3, pp. 402–409, 2020.

[58] D. Wu and J. S. Miguel, "In-stream stochastic division and square root via correlation," in *ACM/IEEE Design Automation Conference*, 2019.

[59] A. Ren, J. Li, Z. Li, C. Ding, X. Qian, Q. Qiu, B. Yuan, and Y. Wang, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," *Acm Sigops Operating Systems Review*, vol. 51, no. 2, pp. 405–418, 2016.

[60] J. Wiart, E. Conil, A. Hadjem, M. Jala, and N. Varsier, "Handle variability in numerical exposure assessment: The challenge of the stochastic dosimetry," in *Antennas and Propagation (EuCAP), 2013 7th European Conference on*, 2013.

[61] S. Xiao, W. Liu, Y. Guo, and Z. Yu, "Low-cost adaptive exponential integrate-and-fire neuron using stochastic computing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2020.

[62] K. Kim, J. Lee, and K. Choi, "Approximate de-randomizer for stochastic circuits," in *2015 International SoC Design Conference (ISOCC)*, 2015.

[63] A. Zell, N. Mache, M. Hüttel, and M. Vogt, "Massive parallelism in neural network simulation," in *Neural Networks, 1993., IEEE International Conference on*, 1993.

[64] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, 2017.

[65] E. Lindholm, "Nvidia tesla: A unified graphics and computing architecture," *IEEE Micro*, vol. 28, 2008.

[66] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[67] "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016.

[68] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *ACM*, 2014.

[69] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *Computer ence*, 2014.

[70] F. Song, S. Tomov, and J. Dongarra, "Enabling and scaling matrix computations on heterogeneous multi-core and multi-gpu systems," in *Acm International Conference on Supercomputing*, 2017.

[71] J. Schemmel, L. Kriener, P. Müller, and K. Meier, "An accelerated analog neuromorphic hardware system emulating nmda- and calcium-based non-linear dendrites," in *International Joint Conference on Neural Networks*, 2017.

[72] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid- State Circuits Conference (ISSCC)*, 2014.

[73] N. R. Mahapatra and B. Venkatrao, "The processor-memory bottleneck," *XRDS: Crossroads, The ACM Magazine for Students*, 1999.

[74] Boahen and K.A, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.

[75] S. C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, "Event-based neuromorphic systems," 2015.

[76] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigné, "Spiking neural networks hardware implementations and challenges: a survey," 2020.

[77] Steve and Furber, "Large-scale neuromorphic computing systems," *Journal of Neural Engineering*, vol. 13, no. 5, pp. 51 001–51 001, 2016.

[78] S. Se, D. G. Lowe, and J. J. Little, "Vision-based global localization and mapping for mobile robots," *IEEE Transactions on Robotics*, 2005.

[79] J. Park, T. Yu, S. Joshi, C. Maier, and G. Cauwenberghs, "Hierarchical address event routing for reconfigurable large-scale neuromorphic systems," *IEEE Transactions on Neural Networks and Learning Systems","pubMedId":"27483491*, 2017.

[80] M. J. Campbell, D. J. Finn, G. K. Tucker, M. Vahey, and R. W. Vedder, "Data-flow multiprocessor architecture with three dimensional multistage interconnection network for efficient signal and data processing," 1991.

[81] G. Indiveri, F. Corradi, and Q. Ning, "Neuromorphic architectures for spiking deep neural networks," *IEEE*, 2016.

[82] H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs, "Fast classification using sparsely active spiking networks," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.

[83] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, and R. K. Krishnamurthy, "A 4096-neuron 1m-synapse 3.8-pj/sop spiking neural network with on-chip stdp learning and sparse weights in 10-nm finfet cmos," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 992–1002, 2019.

[84] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, pp. 82–99, 2018.

[85] Furber, B. S., Galluppi, F., Temple, S., Plana, and A. L., "The spinnaker project," *Proceedings of the IEEE*, 2014.

[86] "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[87] B. V. Benjamin, P. Gao, E. Mcquinn, S. Choudhary, A. R. Chandrasekaran, J. 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, no. 5, pp. 699–716, 2014.

[88] L. Ling, C. T. Pu, and T. Wei, "Continual queries for internet scale event-driven information delivery," *IEEE Transactions on Knowledge Data Engineering*, vol. 11, no. 4, pp. 610–628, 1999.

[89] J. A. Wemmie, J. Chen, C. C. Askwith, A. M. Hruska-Hageman, M. P. Price, B. C. Nolan, P. G. Yoder, E. Lamani, T. Hoshi, and J. H. Freeman, "The acid-activated ion channel asic contributes to synaptic plasticity, learning, and memory." *Neuron*, vol. 34, no. 3, pp. 463–477, 2002.

[90] P. O'Connor and M. Welling, "Deep spiking networks," 2016.

[91] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1-3, pp. 239–255, 2010.

[92] F. H. Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Behind Deep Blue: Building the Computer that Defeated the World Chess Champion, 2002.

[93] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently supporting ad hoc queries in large datasets of time sequences," *ACM SIGMOD Record*, 1997.

[94] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," 2017.

[95] C. Mead, "Analog vlsi and neural systems," *Addison-Wesley Longman Publishing Co., Inc.*, 1989.

[96] R. Silver, K. Boahen, S. Grillner, N. Kopell, and K. L. Olsen, "Neurotech for neuroscience: Unifying concepts, organizing principles, and emerging tools," *Journal of Neuroscience*, vol. 27, no. 44, pp. 11 807–11 819, 2007.

[97] Kwabena and Boahen, "Neurogrid: Emulating a million neurons in the cortex," *Neuroscience Research*, 2010.

[98] L. Liang, "Parallel implementations of hopfield neural networks on gpu," *INRIA-IRISA Rennes Bretagne Atlantique, équipe CAIRN*, 2011.

[99] S. J. Lee, S. U. William, and M. Gerla, "On-demand multicast routing protocol in multihop wireless mobile networks," *Mobile Networks Applications*, vol. 7, no. 6, pp. 441–453, 2002.

[100] S. Joseph, "Neurogrid: Semantically routing queries in peer-to-peer networks," *Springer-Verlag*, 2002.

[101] K. L. Rice, M. A. Bhuiyan, T. M. Taha, C. N. Vutsinas, and M. C. Smith, "Fpga implementation of izhikevich spiking neural networks for character recognition," in *International Conference on Reconfigurable Computing Fpgas*, 2010.

[102] M. R. Azghadi, S. Moradi, D. B. Fasnacht, M. S. Ozdas, and G. Indiveri, "Programmable spike-timing-dependent plasticity learning circuits in neuromorphic vlsi architectures," *Acm Journal on Emerging Technologies in Computing Systems*, vol. 12, no. 2, pp. 1–18, 2015.

[103] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, and S. Millner, "Awafer-scale neuromorphic hardware system for large-scale neural modeling," *IEEE*, 2010.

[104] L. Zhe, R. Ao, L. Ji, Q. Qiu, and Y. Wang, "Structural design optimization for deep convolutional neural networks using stochastic computing," in *2017 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2017.

[105] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, and S. C. Liu, "Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms," *Frontiers in Neuroscience*, vol. 9, no. 222, 2015.

[106] A. V. Schaik, "Building blocks for electronic spiking neural networks," *Neural Networks*, 2001.

[107] K. D. Fischl, G. Tognetti, D. R. Mendat, G. Orchard, J. Rattray, C. Sapsanis, L. F. Campbell, L. Elphage, T. E. Niebur, and A. Pasciaroni, "Neuromorphic self-driving robot with retinomorphic vision and spike-based processing/closed-loop control," in *Information Sciences Systems*, 2017.

[108] A. Mundy, J. Knight, T. C. Stewart, and S. Furber, "An efficient spinnaker implementation of the neural engineering framework," in *International Joint Conference on Neural Networks*, 2015.

[109] H. Zhang, Z. Hu, J. Wei, P. Xie, G. Kim, Q. Ho, and E. Xing, "Poseidon: A system architecture for efficient gpu-based deep learning on multiple machines," *Computer Science*, 2015.

[110] M. Mikaitis, D. R. Lester, D. Shang, S. Furber, and A. Dixius, "Approximate fixed-point elementary function accelerator for the spinnaker-2 neuromorphic chip," in *2018 IEEE 25th Symposium on Computer Arithmetic (ARITH)*, 2018.

[111] S. Davies, J. Navaridas, F. Galluppi, and S. Furber, "Population-based routing in the spinnaker neuromorphic architecture," in *International Joint Conference on Neural Networks*, 2012.

[112] C. Michaelis, A. B. Lehr, W. Oed, and C. Tetzlaff, "Brian2loihi: An emulator for the neuromorphic chip loihi using the spiking neural network simulator brian." 2021.

[113] C. K. Lin, A. Wild, G. Chinya, M. Davies, N. Srinivasa, D. Lavery, and H. Wang, "Programming spiking neural networks on intel's loihi," *Computer*, pp. 1–1, 2018.

[114] M. Potuzak, A. Nichols, D. B. Dingwell, and D. A. Clague, "Hyperquenched volcanic glass from loihi seamount, hawaii," *Earth Planetary Science Letters*, vol. 270, no. 1-2, pp. 54–62, 2008.

[115] B. P. Glackin, T. M. Mcginnity, L. P. Maguire, Q. X. Wu, and A. Belatreche, "A novel approach for the implementation of large scale spiking neural networks on fpga hardware," in *Proceedings of the 8th international conference on Artificial Neural Networks: computational Intelligence and Bioinspired Systems*, 2005.

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

[117] Y. S. Yang and Y. Kim, "Recent trend of neuromorphic computing hardware: Intel's neuromorphic system perspective," in *2020 International SoC Design Conference (ISOCC)*, 2020.

[118] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[119] T. J. Huang, E. C. Science, and P. University, "Imitating the brain with neurocomputer a "new" way towards artificial general intelligence," *International Journal of Automation and Computing*, 2017.

[120] H. D. Garis, "Evolvable hardware genetic programming of a darwin machine," in *International Conference on Artificial Neural Nets Genetic Algorithms*, 1993.

[121] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[122] J. Schemmel, A. Grubl, S. Hartmann, A. Kononov, and M. O. Schwartz, "Live demonstration: A scaled-down version of the brainscales wafer-scale neuromorphic system," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, 2012.

[123] R. Wille, L. Bing, U. Schlichtmann, and R. Drechsler, "From biochips to quantum circuits: computer-aided design for emerging technologies," in *IEEE/ACM International Conference on Computer-aided Design*, 2017.

[124] A. Ankit, A. Sengupta, P. Panda, and K. Roy, "Resparc: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," 2017.

[125] G. W. Burr, P. Narayanan, R. M. Shelby, S. Sidler, and Y. Leblebici, "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)," in *IEEE International Electron Devices Meeting*, 2015.

[126] S. B. Eryilmaz, D. Kuzum, S. Yu, and H. Wong, "Device and system level design considerations for analog-non-volatile-memory based neuromorphic architectures," *IEEE*, 2015.

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

[128] A. M. Zyarah, N. Soures, L. Hays, R. B. Jacobs-Gedrim, and D. Kudithipudi, "Ziksa: On-chip learning accelerator with memristor crossbars for multilevel neural networks," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.

[129] 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.

[130] S. Kim, M. Ishii, S. Lewis, T. Perri, and C. Lam, "Nvm neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning," in *2015 IEEE International Electron Devices Meeting (IEDM)*, 2015.

[131] M. Suri, O. Bichler, D. Querlioz, O. Cueto, and B. Desalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *2011 IEEE INTERNATIONAL ELECTRON DEVICES MEETING (IEDM)*, 2011.

[132] D. Garbin, E. Vianello, O. Bichler, Q. Rafhay, and L. Perniola, "Hfo2-based oxram devices as synapses for convolutional neural networks," *IEEE Transactions on Electron Devices*, vol. 62, no. 8, pp. 2494–2501, 2015.

[133] T. Marukame, K. Nomura, M. Matusmoto, S. Takaya, and Y. Nishi, "Proposal, analysis and demonstration of analog/digital-mixed neural networks based on memristive device arrays," 2018, pp. 1–5.

[134] 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, no. 7550, pp. 61–4, 2015.

[135] W. Dong, D. Lei, T. Pei, M. Cheng, and P. Jing, "Fpga-based neuromorphic computing system with a scalable routing network," in *Non-volatile Memory Technology Symposium*, 2016.

[136] J. Liang and H. S. P. Wong, "Cross-point memory array without cell selectors—device characteristics and data storage pattern dependencies," *IEEE Transactions on Electron Devices*, vol. 57, no. 10, pp. 2531–2538, 2010.

[137] T. Tang, L. Xia, B. Li, W. Yu, and H. Yang, "Binary convolutional neural network on rram," in *Design Automation Conference*, 2017.

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

[139] 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 *Custom Integrated Circuits Conference*, 2011, pp. 1–4.

[140] A. Sapan, Q. Tu-Thach, P. Ojas, A. H. Hsia, E. P. Debenedictis, C. D. James, M. J. Marinella, and J. B. Aimone, "Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding," *Frontiers in Neuroscience*, vol. 9, 2015.

[141] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, 2008.

[142] K. Kim, S. Gaba, D. Wheeler, J. M. Cruzalbrecht, T. Hussain, N. Srinivasa, and W. Lu, "Letter pubs.acs.org/nanolett a functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," 2012.

[143] R. Midya, Z. Wang, S. Asapu, S. Joshi, Y. Li, Y. Zhuo, W. Song, H. Jiang, N. Upadhay, and M. Rao, "Artificial neural network (ann) to spiking neural network (snn) converters based on diffusive memristors," *Advanced Electronic Materials*, 2019.

[144] Z. Wang, S. Joshi, S. Savel'Ev, W. Song, R. Midya, Y. Li, M. Rao, Y. Peng, S. Asapu, and Z. Ye, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electronics*, vol. 1, no. 2, pp. 137–145, 2018.

[145] 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, no. 4, pp. 1297–1301, 2010.

[146] "Emulating short-term and long-term plasticity of bio-synapse based on cu/a-si/pt memristor," *IEEE Electron Device Letters*, vol. 38, no. 9, pp. 1208–1211, 2017.

[147] M. Prezioso, M. R. Mahmoodi, F. M. Bayat, H. Nili, H. Kim, A. Vincent, and D. B. Strukov, "Spike-timing-dependent plasticity learning of coincidence detection with passively integrated memristive circuits," *Nature Communications*, vol. 9, no. 1, 2018.

[148] C. Mesaritakis, A. Kapsalis, A. Bogris, and D. Syvridis, "Artificial neuron based on integrated semiconductor quantum dot mode-locked lasers," *Scientific Reports*, vol. 6, no. 1, p. 39317, 2016.

[149] X. Zhang, Y. Zhuo, Q. Luo, Z. Wu, R. Midya, Z. Wang, W. Song, R. Wang, N. K. Upadhyay, and Y. Fang, "An artificial spiking afferent nerve based on mott memristors for neurorobotics," *Nature Communications*, 2020.

[150] H. Jayakumar, A. Raha, Y. Kim, S. Sutar, W. S. Lee, and V. Raghunathan, "Energy-efficient system design for iot devices," in *Asia South Pacific Design Automation Conference*, 2016, pp. 298–301.

[151] F. Akopyan, "Design and tool flow of ibm's truenorth: an ultra-low power programmable neurosynaptic chip with 1 million neurons," in *the 2016*, 2016.

[152] S. Sato, K. Nemoto, S. Akimoto, M. Kinjo, and K. Nakajima, "Implementation of a new neurochip using stochastic logic," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1122–1127, 2003.

[153] L. I. Hongge, Y. Hayakawa, S. Sato, and K. Nakajima, "Hardware implementation of an inverse function delayed neural network using stochastic logic," *IEICE - Transactions on Information and Systems*, vol. E89-D, no. 9, pp. 2572–2578, 2006.

[154] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–16, 2020.

[155] S. A. Mansouri, D. Gallear, and M. H. Askariazad, "Decision support for build-to-order supply chain management through multiobjective optimization," *International Journal of Production Economics*, vol. 135, no. 1, pp. 24–36, 2012.

[156] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "Vlsi implementation of deep neural network using integral stochastic computing," *IEEE Transactions on Very Large Scale Integration Systems*, vol. PP, no. 10, pp. 1–12, 2017.

[157] L. Zhe, L. Ji, A. Ren, R. Cai, and Y. Wang, "Heif: Highly efficient stochastic computing-based inference framework for deep neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2018.

[158] L. Ji, Z. Yuan, L. Zhe, R. Ao, and Y. Wang, "Normalization and dropout for stochastic computing-based deep convolutional neural networks," *Integration the VLSI Journal*, vol. 65, 2017.

[159] Q. Qiu, Z. Li, C. Ding, X. Qian, A. Ren, Y. Wang, J. Li, and B. Yuan, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," *Computer architecture news*, 2017.

[160] Da, Zhang, Hui, and Li, "A stochastic-based fpga controller for an induction motor drive with integrated neural network algorithms," *Industrial Electronics IEEE Transactions on*, 2008.

[161] T. Hirtzlin, B. Penkovsky, M. Bocquet, J. O. Klein, J. M. Portal, and D. Querlioz, "Stochastic computing for hardware implementation of binarized neural networks," *IEEE Access*, vol. 7, pp. 76 394–76 403, 2019.

[162] L. Zhe, R. Ao, L. Ji, Q. Qiu, and Y. Bo, "Dscnn: Hardware-oriented optimization for stochastic computing based deep convolutional neural networks," in *IEEE International Conference on Computer Design*, 2016.

[163] C. Soize, "Stochastic models of uncertainties in computational structural dynamics and structural acoustics," *Springer Vienna*, 2012.

[164] H. Xiong, M. A. Bakar, and G. He, "Hardware implementation of an improved stochastic computing based deep neural network using short sequence length," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2020.

[165] H. Wang, Z. Zhang, X. You, and C. Zhang, "Low-complexity winograd convolution architecture based on stochastic computing," in *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, 2018.

[166] D. Neil and S. C. Liu, "Minitaur, an event-driven fpga-based spiking network accelerator," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 22, no. 12, pp. 2621–2628, 2014.

[167] 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 *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.

[168] C. Zhao, B. T. Wysocki, C. D. Thiem, N. R. Mcdonald, and Y. Yang, "Energy efficient spiking temporal encoder design for neuromorphic computing systems," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 4, pp. 265–276, 2017.

[169] Y. Chen, L. Tao, S. Liu, S. Zhang, and O. Temam, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*.

[170] Yu-Hsin, Chen, Tushar, Krishna, Joel, S., Emer, Vivienne, and Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid State Circuits*, 2017.

[171] Ardavan, Pedram, Song, Han, Xingyu, Liu, Jing, Pu, William, and J., "Eie: Efficient inference engine on compressed deep neural network," *Computer Architecture News*, 2016.

[172] G. Yilma, Z. Qin, M. Assefa, G. Alemu, and M. Ayalew, "Attention augmented convolutional neural network for fine-grained plant disease classification and visualization using stochastic sample transformations," 2021.

[173] Z. Kuang, J. Wang, S. Yang, G. Yi, and X. Wei, "Digital implementation of the spiking neural network and its digit recognition," in *2019 Chinese Control And Decision Conference (CCDC)*, 2019.

[174] M. Alawad, H. J. Yoon, and G. Tourassi, "[ieee 2017 ieee international conference on big data (big data) - boston, ma, usa (2017.12.11-2017.12.14)] 2017 ieee international conference on big data (big data) - energy efficient stochastic-based deep spiking neural networks for sparse datasets," pp. 311–318, 2017.

[175] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *IEEE International Conference on Computer Design*, 2013.

[176] Brown, D. Bradley, Card, and C. Howard, "Stochastic neural computation i: Computational elements." *IEEE Transactions on Computers*, 2001.

[177] Gerstner, Wulfram, Kistler, and M. Werner, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Spiking Neuron Models, 2002.

[178] S. Maya, R. Reynoso, C. Torres-Huitzil, and M. Arias-Estrada, "Compact spiking neural network implementation in fpga," in *International Workshop on Roadmap to Reconfigurable Computing*, 2000.

[179] L. Y. Niu, Y. Wei, J. Y. Long, and W. B. Liu, "High-accuracy spiking neural network for objective recognition based on proportional attenuating neuron," *Neural Processing Letters*, pp. 1–19, 2022.

[180] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. V. Schaik, R. Etienne-Cummings, T. Delbruck, S. C. Liu, P. Dudek, P. H?Fliger, and S. Renaud, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, no. 73, p. 73, 2011.

[181] M. J. Pearson, C. Melhuish, A. G. Pipe, M. Nibouche, and B. Mitchinson, "Design and fpga implementation of an embedded real-time biologically plausible spiking neural network processor," in *International Conference on Field Programmable Logic Applications*, 2005.

[182] J. Liu, Z. Liang, Y. Luo, J. Huang, and S. Yang, "Hardware tripartite synapse architecture based on stochastic computing," in *2019 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, 2019.

[183] R. Kumar, "Stochastic computing: Embracing errors in architecture and design of hardware and software," 2011.

[184] Y. Horio, T. Taniguchi, and K. Aihara, "An asynchronous spiking chaotic neuron integrated circuit," *Neurocomputing*, vol. 64, no. 1, pp. 447–472, 2005.

[185] H. Jie and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Test Symposium (ETS), 2013 18th IEEE European*, 2013.

[186] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, "Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework," *Neural Netw*, pp. 49–58, 2018.

[187] C. Ding, S. Liao, Y. Wang, L. Zhe, L. Ning, Y. Zhuo, W. Chao, X. Qian, B. Yu, and Y. Geng, "C ir cnn: accelerating and compressing deep neural networks using block-circulant weight matrices," in *IEEE/ACM International Symposium on Microarchitecture*, 2017.

[188] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, 2016.

[189] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *European Conference on Computer Vision*, 2016.

[190] E. O. Adewuyi, E. K. O'Brien, D. R. Nyholt, T. Porter, and S. M. Laws, "A large-scale genome-wide cross-trait analysis reveals shared genetic architecture between alzheimer's disease and gastrointestinal tract disorders," *Communications Biology*, vol. 5, no. 1, 2022.

[191] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, and G. Y. Wei, "14.3 a 28nm soc with a 1.2ghz 568nj/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for iot applications," in *International Symposium on Solid State Circuits (ISSCC)*, 2017.

[192] N. Rathi, P. Panda, and K. Roy, "Stdp based pruning of connections and weight quantization in spiking neural networks for energy efficient recognition," *arXiv e-prints*, 2017.

[193] S. Sen, S. Venkataramani, and A. Raghunathan, "Approximate computing for spiking neural networks," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017.

[194] E. O. Neftci, B. U. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs, "Stochastic synapses enable efficient brain-inspired learning machines," 2015.

[195] G. Srinivasan, A. Sengupta, and K. Roy, "Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip stdp learning," *Rep*, vol. 6, p. 29545, 2016.

[196] S. Sheik, S. Paul, C. Augustine, C. Kothapalli, and E. Neftci, "Synaptic sampling in hardware spiking neural networks," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016.

[197] KnagPhil, LuWei, and ZhangZhengya, "A native stochastic computing architecture enabled by memristors," *IEEE Transactions on Nanotechnology*, 2014.

[198] M. Jerry, A. Parihar, B. Grisafe, A. Raychowdhury, and S. Datta, "Ultra-low power probabilistic imt neurons for stochastic sampling machines," in *2017 Symposium on VLSI Circuits*, 2017.

[199] C. Frenkel, J. D. Legat, and D. Bol, "A 0.086-mm^2 9.8-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28nm cmos," 2018.

[200] "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, 2015.

[201] S. Oskouei, H. Golestani, M. Kachuee, M. Hashemi, H. Mohammadzade, and S. Ghiasi, "Gpu-based acceleration of deep convolutional neural networks on mobile platforms," *Computer Science*, 2015.

[202] W. El-Harouni, S. Rehman, B. S. Prabakaran, A. Kumar, and M. Shafique, "Embracing approximate computing for energy-efficient motion estimation in high efficiency video coding," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017.

[203] Y. Kim, Z. Yong, and L. Peng, "An energy efficient approximate adder with carry skip for error resilient neuromorphic vlsi systems," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, 2013.

[204] L. Lu, L. Yun, Q. Xiao, and S. Yan, "Evaluating fast algorithms for convolutional neural networks on fpgas," in *IEEE International Symposium on Field-programmable Custom Computing Machines*, 2017.

[205] M. Mathieu, M. Henaff, and Y. Lecun, "Fast training of convolutional networks through ffts," *Eprint Arxiv*, 2013.

[206] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. Lecun, "Fast convolutional nets with fbfft: A gpu performance evaluation," 2014.

[207] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[208] J. Cong and B. Xiao, "Minimizing computation in convolutional neural networks," in *International Conference on Artificial Neural Networks*, 2014.

[209] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," *IEEE*, 2015.

[210] A. L. Toom, "The complexity of a scheme of functional elements realizing the multiplication of integers," *Dokl.akad.nauk Sssr*, vol. 3, no. 3, p. 496ndash;498, 1963.
[211] S. A. Cook and S. O. Aanderaa, "On the minimum computation time of functions," *Transactions of the American Mathematical Society*, vol. 142, pp. 291–314, 1969.
[212] Winograd and Shmuel, "Arithmetic complexity of computations ‖ 7. cyclic convolution and discrete fourier transform," vol. 10.1137/1.9781611970364, pp. 71–92, 1980.
[213] D. Kappel, "Synaptic sampling: A bayesian approach to neural network plasticity and rewiring," in *International Conference on Neural Information Processing Systems-volume*, 2015.
[214] Peter, U., Diehl, Matthew, and Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, p. 99, 2015.
[215] K. V. Sickle and H. Abdel-Aty-Zohdy, "A reconfigurable spiking neural network digital asic simulation and implementation," *IEEE*, 2009.
[216] S. Chaturvedi, A. A. Khurshid, and S. S. Dorle, "Reconfiguration of spiking neural network for optimization with applications to image processing," in *International Conference on Emerging Trends in Engineering Technology*, 2014.
[217] D. Molka, D. Hackenberg, R. Schne, and M. Müller, "Igcc 2010 conference presentation: Characterizing the energy consumption of data transfers and arithmetic operations on x86-64 processors," 2016.
[218] B. Belhadj, A. Valentian, P. Vivet, M. Duranton, and O. Temam, "The improbable but highly appropriate marriage of 3d stacking and neuromorphic accelerators," *ACM*, 2014.
[219] K. Chang, D. Kadetotad, C. Yu, J. S. Seo, and S. K. Lim, "Monolithic 3d ic designs for low-power deep neural networks targeting speech recognition," in *IEEE/ACM International Symposium on Low Power Electronics Design*, 2017.
[220] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory," *IEEE Press*, 2016.
[221] C. Bao, S. K. Seol, and W. S. Kim, "A 3d integrated neuromorphic chemical sensing system," *Sensors and Actuators B Chemical*, vol. 332, p. 129527, 2021.
[222] M. A. Ehsan, H. An, Z. Zhou, and Y. Yi, "A novel approach for using tsvs as membrane capacitance in neuromorphic 3-d ic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1640–1653.
[223] P. Batude, C. Fenouillet-Beranger, L. Pasini, V. Lu, and M. Vinet, "3dvlsi with coolcube process: An alternative path to scaling," in *Vlsi Technology*, 2015.
[224] K. Sakuma, "3d integration in vlsi circuits: Implementation technologies and applications," 2018.
[225] Morro, Canals, Oliver, M. L, Alomar, Galan-Prado, P. J, Ballester, J. L, and Rossello, "A stochastic spiking neural network for virtual screening." *IEEE transactions on neural networks and learning systems*, 2017.
[226] A. Yousefzadeh, G. Orchard, E. Stromatias, T. Serrano-Gotarredona, and B. Linares-Barranco, "Hybrid neural network, an efficient low-power digital hardware implementation of event-based artificial neural network," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018.

LU CHEN received the B.S. degrees in biomedical engineering from The Sichuan University of Science and Engineering, Zigong, China, in 2016 and 2020. She was admitted to Sichuan University of Science and Engineering in 2020, where he is currently studying the M.S. degree. Her main research direction is Intelligent Chip

XINGZHONG XIONG received the B.S. degree in communication engineering from the Sichuan University of Science and Engineering (SUSE), Zigong, China, in 1996, and the M.S and Ph.D. degrees in communication and information system from the University of Electronic Science and Technology of China (UESTC), in 2006 and 2009, respectively.

He is currently a Professor with the School of Automation and Electronic Information, SUSE. His research interests include wireless communications and intelligent signal processing.

JUN LIU received the B.S. and M.S. degrees in communication engineering and pattern recognition and intelligent systems from the Sichuan University of Science and Engineering, Zigong, China, in 2017 and 2020, respectively. From 2018 to 2020, he has been a Visiting Scholar with the Department of Aeronautics and Aeronautics, Shanghai Jiao Tong University (SJTU). Since 2021, he has been with the Sichuan University of Science and Engineering, where he is currently a Lecturer. His research interests include signal processing for wireless communication, intelligent signal processing, machine learning and pattern recognition, computer vision, and information fusion.