IEEE Journal on Exploratory Solid-State Computational Devices and Circuits

Received 31 July 2019; revised 16 October 2019 and 11 November 2019; accepted 22 November 2019.
Date of publication 27 November 2019; date of current version 2 January 2020.

Digital Object Identifier 10.1109/JXCDC.2019.2956112

Benchmarking Delay and Energy of Neural Inference Circuits

DMITRI E. NIKONOV (Senior Member, IEEE) and IAN A. YOUNG (Fellow, IEEE)

Components Research, Intel Corporation, Hillsboro, OR 97124 USA

CORRESPONDING AUTHOR: D. E. NIKONOV (dmitri.e.nikonov@intel.com)

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

ABSTRACT Neural network circuits and architectures are currently under active research for applications to artificial intelligence and machine learning. Their physical performance metrics (area, time, and energy) are estimated. Various types of neural networks (artificial, cellular, spiking, and oscillator) are implemented with multiple CMOS and beyond-CMOS (spintronic, ferroelectric, and resistive memory) devices. A consistent and transparent methodology is proposed and used to benchmark this comprehensive set of options across several application cases. Promising architecture/device combinations are identified.

INDEX TERMS Benchmarking, beyond-CMOS, CNN, neural network, neuromorphic, power, spiking, spintronic, throughput.

I. INTRODUCTION

THE unprecedented progress of traditional Boolean computing over the last five decades has been propelled by the scaling of the transistor according to Moore's law [1]. Recently, a large share of computing is being consumed by applications related to artificial intelligence (AI) and machine learning (ML). For these, Boolean computing is less efficient. This has spurred research in neural computing that covers a wide field of research, from neural network algorithms that can be programmed on traditional Boolean hardware, such as CPUs or GPUs, to neural network circuits implemented in specialized hardware—application-specific engines. The former approach currently handles the majority of user needs from the data center to the edge. The latter approach resulted in both development and research thrusts in, e.g., digital neural accelerators, such as [2] (see a review [3]) and neuromorphic (biologically inspired) chips, such as [4]. The operation of neuromorphic chips can span a range of circuit implementations from mostly digital to mostly analog (see reviews [5] and [6]).

In the last few years, AI/ML achieved prominent successes, especially related to deep neural networks (DNNs) [7] and convolutional neural networks (CoNNs). ML has enabled a revolutionary improvement in the accuracy of image, pattern, and facial recognition, including the treatment of ''big data'' online. More demand for neural computing is emerging in robotic control, autonomous vehicles, drones, and so on.

One of the main concerns on the minds of developers of AI computing systems is the same as for traditional computing: the power consumption in the chips. The history of traditional computing shows that the commercial success of computing devices and architectures is predicated largely on their physical performance—areal density, speed of operation, and consumed energy, as benchmarked in [8] and [9]. These ultimately translate into processing throughput and consumed power of the chips that are of utmost importance to the user. A fair comparison between the published neural network implementations is difficult due to the difference in the process technology generation, the network architectures, and computing workloads.

The main purpose of this article is to establish a methodology for comparing various neural network hardware approaches and to understand the trends revealed through its development. In doing that, we strive to adhere to the following principles.

1) *General:* Wide scope of technologies, devices, and circuits;
2) *Transparent:* Simple analytics more important than precise simulations;
3) *Uniform:* Consistent inputs and assumptions across multiple types of hardware;
4) *Reproducible:* All models used are described, and the code is available [10] to the reader for verification.

Let us differentiate this work from the existing body of literature. We do not aim to give a literature review and refer the reader to the excellent review articles in the neuromorphic hardware field [11]–[13], which do not attempt to quantitatively compare prior works, as we do in this article. Oftentimes, benchmarking refers to comparing various algorithms for an application mainly based on their accuracy and with little reference to hardware implementation, e.g., [14]. In contrast, we compare different types of hardware implementing the same algorithm and focusing on its energy consumption and performance. The discussion of neural networks in the

articles published is just focused at the architecture level. For example, the accuracy of recognition is studied in its dependence on the number of network elements, topology, and details of the algorithm. We are cognizant of the importance of the accuracy of inferencing, and indeed, prior studies discovered that this accuracy can be degraded compared to the algorithm-limited "maximum accuracy" due to device nonidealities [15]. However, for this article, we focus on the effect of the different types of devices and their neural circuits. For that purpose, we make an optimistic assumption that the devices are not degrading their characteristics (and accuracy) with the operation, as exemplified by [16]. This assumption is appropriate for our benchmarking that targets the idealized, paradigm limiting cases. Some research articles report experimentally measured performance and energy for several implementations of CoNN [17], [18] by running them on a GPU or a particular DNN implemented on a variety of neural hardware (top–down) [19]. In contrast, we provide a theoretical prediction approach for benchmarking (bottom–up). A rigorous simulation framework, NeuroSim, is used to benchmark the neural network circuit architecture in a cross-connect topology based on a set of memory cells serving as synapses [20]. The Eyeriss simulation tool focuses on accelerators for DNN [21]. The CrossSim simulator has similar capabilities [22]. Benchmarking for a variety of device technologies, including exploratory beyond-CMOS ones, has been done in an approach similar to ours, but for only one type of neural network, i.e., cellular neural networks (CeNNs) [23]. Estimates of time and energy of operation for certain types of digital and analog devices have been done previously [24]–[27]. Compared to these prior works, we cover a wider scope of devices and network architectures, using less detailed, but adequate, circuit models.

Another purpose of this article is to explore the impact of exploratory devices on the performance and energy efficiency of operation for neural networks. For example, neural circuits based on the spintronic type of beyond-CMOS devices have been proposed [28], [29]. In this article, we aim to expand the list of beyond-CMOS devices applied to neural networks. Also, we consider both digital and analog neural networks in an attempt to understand whether there is an advantage in the speed and energy of neural computing implemented with the latter. We also consider several types of neural network microarchitectures and analyze their relative advantages. Finally, we consider several cases of neural networks running their application "workloads" and demonstrate that the qualitative conclusions made about various neural network hardware remain valid when performing neural computing with these "real use" cases.

II. FUNDAMENTALS AND CONCEPTS OF NEUROMORPHIC COMPUTING

Operation in the majority of neural network architectures relies on a neural gate, often called the perceptron [3]. The elements at the input, synapses, receive vectors of input signals x_i and multiply them by vectors of weights w_i. Neurons perform the summation of these products and apply a nonlinear threshold (or "activation") function

$$f(x) = g\left(\sum_{i}^{n} w_i x_i + b\right). \tag{1}$$

Despite its apparent simplicity, the neural gate in some forms underlies most of the neuromorphic hardware and algorithms. DNNs consist of cascaded multiple layers of neural gates. CoNNs are an example of algorithms applying DNN to image processing. For the benchmarking analysis of this article, we only consider the DNN workloads of inference (i.e., determining a distance of input vectors from memorized ones in multidimensional space). The inference is crucial for recognition, i.e., classifying objects in the input data.

Learning (or training) is the process of modifying parameters of the neural network for better recognition. It consists of performing numerous inferences on the input data and then adjusting the weights in the neural network. Methods of learning can be, for example: 1) supervised learning—optimization of weights through, e.g., backpropagation algorithm [7] or 2) unsupervised learning—change of weights according to synapse activity caused by input patterns, e.g., using the spike-timing-dependent plasticity (STDP) algorithm [28].

We decided to limit the scope of this article to inferencing. We realize the importance of learning and that it requires much more computing effort. Also, inference and learning present different market segments and have different usage models; learning is mostly practiced by providers of data center services and inference mostly client users. As such, it is possible to consider inference and learning separately. Benchmarking of learning (such as in [20]) will be explored in a future publication.

III. TYPES OF NEUROMORPHIC DEVICES

Synapses and neurons can be implemented by a variety of devices [27]: digital CMOS and analog CMOS or tunnel FET (TFET) devices; ferroelectric FET (FEFET) devices; and spintronic devices [28] of five types: in-plane and perpendicular spin–transfer torque (STT) switches with perpendicular magnetic anisotropy, spin–orbit torque (SOT) switches, domain-wall (DW) motion devices, and magnetoelectric (ME) switched devices. Resistive memory elements include oxide RRAM, floating gate resistors (flash), phase-change memory, general spintronic and higher-resistance spin-orbit torque resistors [implemented as magnetic tunnel junctions (MTJs)], and ferroelectric resistors (FTJ), see Figs. 1–3. To determine the area, delay, and energy of devices and circuits, we rely on our benchmarking methodology [8], [9] that was developed for digital circuits. The benchmarks are calculated consistently for multiple devices scaled to the process node size [8], $F = 15$ nm. We first determine the values for an "intrinsic device" (a transistor or a nanomagnet [30]) and then for simple circuits [9]. For digital technologies, we assume that the synapses are comprised of 8-bit registers. For analog technologies, we will assume that the synapses are accurately set to 64 levels. Understanding that these two are not equivalent in terms of precision, we choose to keep the typical value of 8 for digital precision. We also assume that the accuracy in analog networks is not limited by the number of levels. We take the best case by ignoring nonidealities of the synapse device characteristics. Realistic cases in which device characteristics affect the accuracy are considered, e.g., in [31].

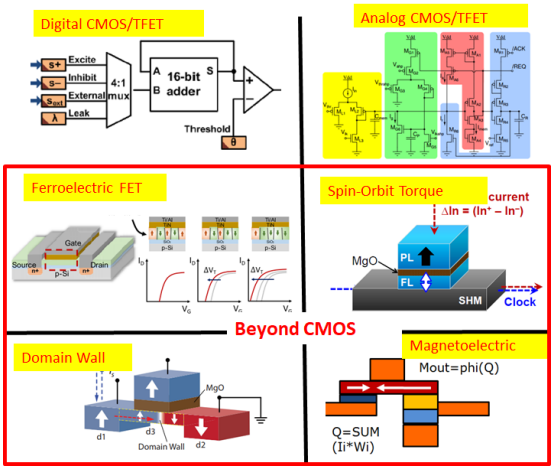

FIGURE 1. **Neurons implemented with CMOS and beyond-CMOS devices.**

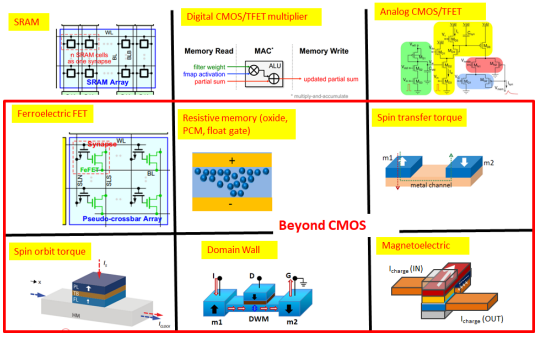

FIGURE 2. **Synapses implemented with CMOS and beyond-CMOS devices.**

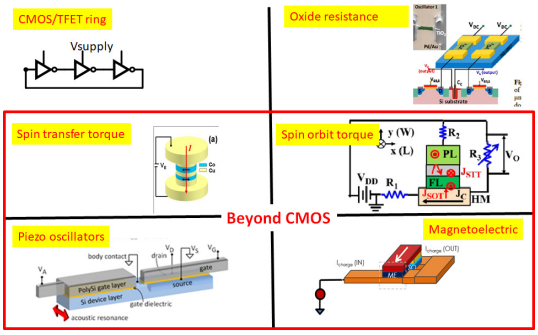

FIGURE 3. **Schemes of oscillators used as synapses.**

IV. TYPES OF NEURAL NETWORKS

We classify neural networks into four types according to the nature of signals used.

1) Artificial neural networks (ANNs) where the outputs switch in response to inputs in a mostly monotonic fashion.
2) CeNNs differ from ANN by their rectangular grid geometry and high connectivity. Here, they are treated in agreement with [23] based on the methodology in [9].
3) Spiking neural networks (SNNs) receive trains of spikes at inputs. Synapses route spikes toward neurons, and neurons fire output spikes depending on their input spike timing.
4) Oscillator neural networks (ONNs) can approximate convolutions by the analog output resulting from the synchronization of oscillators.

Neural networks can be created with various combinations of neurons and synapses [30, Table 2]. To describe these, the first letter of the label (A, C, S, and O) designates the type of the neural network, the second letter designates the type of the neuron, the third letter designates the type of the synapse, and the last letter designates whether the network is digital or analog. In neural networks, the synapse and neuron circuits require tens of transistors. Alternatively, single-spintronic devices are capable of implementing synapses and neurons [28]. We will use the same synapses across the ANN, CeNN, and SNN classes though neurons will be different.

V. TREATMENT OF INTERCONNECTS

The benchmarks for neural network elements, neural gates, and larger DNNs are built up hierarchically, from benchmarks for a synapse and a neuron obtained in Section IV. We refer to it as "bottom–up benchmarking."

The functional chip comprises a number of neural cores with multiple neurons in each and multiple synapses feeding signals into each core. The total number of synapses per chip is thus

$$s_{ch} = c_{ch} n_{cor} s_{neu}. \tag{2}$$

All notations are defined in Table 1. Empirical factors are introduced to account for layout overhead: spacing between circuits for interconnects, routing circuits, intermediate registers, encoders/decoders, and so on. To obtain the corrected area, the estimated area is multiplied by the additional layout overhead factors (see Table 1). For a certain workload, only a share of synapses r_a may be active. The area of the chip is then

$$a_{ch} = M_{ch} c_{ch} (M_{cor} n_{cor} (M_{neu} a_{neu} + s_{neu} M_{syn} a_{syn})). \tag{3}$$

The operating energy of an interconnect is mainly determined by the interconnect capacitance per unit length. The energy to charge an interconnect is

$$E_{ic} = c_{ic} l V^2 \tag{4}$$

where the length of an interconnect from a circuit block to the next block is calculated as

$$l = \sqrt{a_{circ}}. \tag{5}$$

The area of the relevant circuit block for synapses $a_{circ,syn} = a_{syn} s_{cor}$ is set by the requirement to deliver the synapse output within the area of the core, and for neurons $a_{circ,neu} = a_{ch}$, it is set by the requirement to deliver, i.e., interconnect, the output signal of a neuron to any part of the chip.

VI. CHIP-LEVEL BENCHMARKS

The operation of the chip involves signals coming from input neurons, processed in synapses, and then firing of output neurons. A synaptic operation (synaptic event) is understood in nonspiking networks as an operation of multiplication of an input signal by weight, i.e., "multiply-and-accumulate" (MAC). However, in spiking networks, a synaptic event is mostly understood as "a spike event is a synaptic operation

TABLE 1. List of notation used in this article.

Quantity	Symbol	Units	Value
Process generation ('node') size	F	nm	15
Minimum interconnect length	l_{ic}	nm	20F
Bits in a digital synapse	n_b		8
Levels in an analog synapse	n_l		64
Area, delay, energy of circuits	a,τ,E	nm^2,ps,fJ	
Area, delay, energy of a device	$a_{dev},\tau_{dev},E_{dev}$	nm^2,ps,fJ	
Area, delay, energy of a minimal interconnect	a_{ic},τ_{ic},E_{ic}	nm^2,ps,fJ	
Area, delay, energy of an inverter	$a_{inv},\tau_{inv},E_{inv}$	nm^2,ps,fJ	
Area, delay, energy of a 2 input NAND	$a_{nan},\tau_{nan},E_{nan}$	nm^2,ps,fJ	
Area, delay, energy of a register bit	$a_{reg},\tau_{reg},E_{reg}$	nm^2,ps,fJ	
Area, delay, energy of a state element	a_{se},τ_{se},E_{se}	nm^2,ps,fJ	
Area, delay, energy of a 1-bit full adder	a_1,τ_1,E_1	nm^2,ps,fJ	
Area, delay, energy of a n-bit full ripple-carry adder	$a_{add},\tau_{add},E_{add}$	nm^2,ps,fJ	
Area, delay, energy of a synapse	$a_{syn},\tau_{syn},E_{syn}$	nm^2,ps,fJ	
Area, delay, energy of a neuron	$a_{neu},\tau_{neu},E_{neu}$	nm^2,ps,fJ	
Width of a digital transistor	w_{dt}	nm	4F
Width of an analog transistor	w_{at}	nm	16F
Capacitance of the transistor per unit width	c_{tran}	F/m	
On- and off-current in the transistor per unit width	i_{on},i_{off}	A/m	
Saturation voltage of a transistor	V_{sat}	V	0.3
Linear transconductance of a transistor	g_{mdt}	S	
On-state resistance of a transistor	R_{ondt}	Ω	
Supply voltage	V_{cc}	V	0.8
Capacitance of an interconnect per length	c_{ic}	nF/m	0.5
Capacitance of a minimum interconnect	C_{ic}	F	$c_{ic}l_{ic}$
Resistance of an interconnect per length	r_{ic}	$G\Omega$	2.2
Resistance of a minimum interconnect	R_{ic}	Ω	667
Load capacitance for an interconnect	C_{load}	F	
Effective resistance of a synapse	R_{eff}	Ω	
Voltage for a sense amplifier	V_{sa}	V	0.4
Transistor width for a sense amplifier	$w_{p,n,iso,en}$	nm	{4,4,6.5,5}F
Sense difference and cell read voltages for a voltage sense amplifier	V_{vsa},V_{rvsa}	V	{0.1,0.5}
Analog read circuit row voltage	V_{row}	V	0.65
Analog read pulse	τ_{repu}	ns	1
Width of input, pull-up, output transistors in OTA	$w_{in,up,out}$	nm	{10,5,10}F
Current from a neuron	I_{neu}	A	
Factor of extra number of synapses in CeNN	M_{syncnn}		4
Factor of extra settling time in CeNN	$M_{stepcnn}$		5
Maximum weight value for CeNN	$w_{\max}$		0.23
Average sum of the weights in CeNN cell	w_{sum}		1.26
Factors of spike duration and spacing between spikes	N_{spi},N_{spa}		3
Number of spikes for a neuron to fire	N_{fire}		10
Ratio of active synapses	r_a		
Number of oscillator periods till synchronization	N_{synch}		30
Area overhead factor for a synapse, neuron	M_{syn},M_{neu}		2
Area overhead factor for a core, chip	M_{cor},M_{ch}		2
Cores per chip; value for a nominal chip	c_{ch}		64
Neurons per core; value for a nominal chip	n_{cor}		256
Number of input and output neurons per core	n_{in},n_{out}		
Synapses per neuron	s_{neu}		256
Synapses per core, per chip	s_{cor},s_{ch}		
Number of feature maps in a stage	f_{st}		
Throughput of synaptic operations	T_{syn}	s^{-1}	

evoked when one action potential is transmitted through one synapse'' according to [32]. There may be multiple spikes required to fire a neuron, i.e., multiple synaptic operations correspond to one MAC. Therefore, the firing rate is conventionally defined differently for spiking, SNN (to keep it consistent with definitions in [4]), and nonspiking, ANN, CeNN, and ONN, networks

$$f_{fire} = 1/\tau_{syn} \text{ (nonspiking)} \tag{6}$$

$$f_{fire} = 1/(r_a s_{neu}\tau_{syn}) \text{ (spiking)}. \tag{7}$$

The time step in a chip corresponds to the operation of one stage of a neural network. It consists of the time for enough synaptic inputs to arrive at the neuron to make it fire plus the delay in the neuron itself

$$\tau_{step} = 1/f_{fire} + \tau_{neu}. \tag{8}$$

Total energy per synaptic event contributed by a synapse and a share of neuron energy is

$$E_{syntot} = E_{syn} + E_{neu}/(r_a s_{neu}). \tag{9}$$

Publications normally quote the throughput of synaptic operations per second (SOPS) (not to be confused with a throughput of inferences, as given in the following)

$$T_{syn} = f_{fire} r_a s_{ch}. \tag{10}$$

The dissipated power and the energy per time step are

$$P_{ch} = T_{syn}E_{syntot} \quad E_{ch} = P_{ch}\tau_{ch}. \tag{11}$$

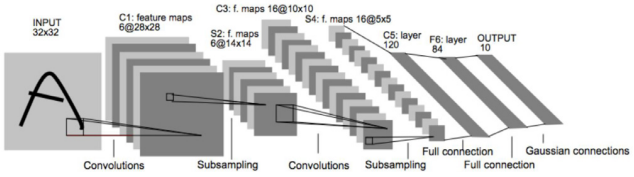

FIGURE 4. Map of layers in the CoNN, LeNet [33].

To compare the benchmarks between actual chips and bottom–up estimates for various neural networks, we calculate the performance of the latter for a nominal chip with parameters given in [30].

VII. NEUROMORPHIC WORKLOADS AND HARDWARE

We have considered examples of neuromorphic workloads, including CoNN, such as LeNet [33], [34] (see Fig. 4), AlexNet [35], a single-stage convolution of a 5×5 pixel fragment with 24 filters, a single-stage associative memory of pixel patterns [23], a DNN for recognition of handwritten digits from the MNIST handwritten-digit image database [34] implemented as a multilayer perceptron (MLP) with $784 \times 256 \times 128 \times 10$ fully connected neurons in layers, and a DNN for speech recognition from [19]—a four-layer MLP with $390 \times 256 \times 256 \times 29$ neurons. While all of these networks belong to the class of nonrecurrent DNN, these are examples of ubiquitous applications required by users. However, these workloads may not be favorable to SNNs. For example, they do not utilize temporal information carried by spikes. The reader should be warned that results may change if we consider workloads more favorable to SNNs. Now, we determine the benchmarks for a part of the chip necessary to perform a specific computing workload (we will use the subscript CW). More specifically, in our case, the computing workload is an inference. Its hardware implementation is determined by the logic structure of a neural network, which can be thought of as an algorithm. Each feature map in a stage is produced by convolution with one of the kernels; this process is mapped to a neural core. The number of neurons and synapses in each core is determined by the connectivity of the neural network. Each core will have a number of input neurons and a number of output neurons. Often overlooked input neurons are shown in the array schemes, e.g., in [4]. Each output neuron collects inputs from the number of active synapses per neuron. For example, the LeNet NN shown in [30] can be implemented by an application-specific design comprising a set of neural cores, Fig. 5. A general-purpose neural chip is composed of cores of a fixed size with some of the input and output neurons remaining unused.

The total number of synapses in the DNN can be very large. It is much larger than the number of trainable weights (which can be, e.g., filters for the feature maps in CoNN). In this benchmark, we adopt an approach of multiple copies of weights written into the memory of synapses and placing them close to neurons. This requires a larger number of memory cells and the corresponding chip area dedicated to them. However, it can be affordable in the case of dense analog memory. During training, this requires more time and energy for updating the weights. The alternative—using only the necessary memory to hold trainable parameters—has the major downside of the need to route connections

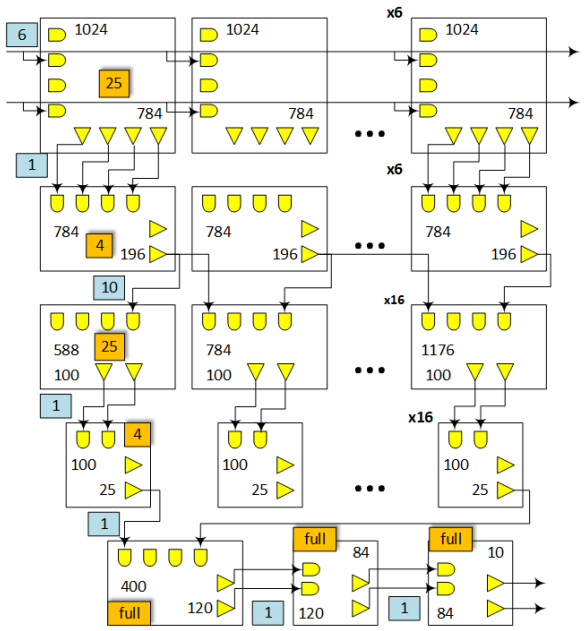

FIGURE 5. Block diagram of an implementation of the CoNN, LeNet. "AND" symbols designate input neurons and triangles designate output neurons; their number of instantiations is indicated. Numbers in orange squares designate the number of synapses per neuron in the respective core; "full" means a fully connected core. Numbers in blue squares next to buses connecting cores designate the fan-out for output neurons.

with numerous neurons and the time and energy to fetch the weight values.

Focusing on the structure of a core, we envision that different topologies can be chosen for interconnecting input and output neurons by synapses. The most straightforward one is the cross connect [30]. It is best for fully connected layers (such as those in the bottom part of Fig. 5) and allows for a general pattern of connections. However, it will leave many unused synapses in case of a sparsely connected NN. Since the processing of information is happening in or on the periphery of the memory array, containing the weights, such a scheme, can also be classified as "compute-in-memory" or "inference-in-memory." The convolution topology [30] is specifically designed for connections in convolution layers (such as those in the top part of Fig. 5). This scheme utilizes the property of CoNN—sparse connectivity between neurons. It also closely resembles cellular NN (CeNN) connectivity. In this topology, the output neurons are placed close to the connected input ones. They mimic the positions of pixels in the image and the resulting feature map. The convolution topology is efficient since it contains only active and no unused synapses. However, it is not general—additional synapses need to be designed for less sparse connectivity.

The means by which long interconnects can be routed is shown in Fig. 6. For the cross-connect topology, the routing is trivial since both input and output neurons are at the edge of the core. Even for the convolution topology, the input and output wires can enter in a regular array of wires and still be routed to neurons. The pitch of interconnect wires is assumed to be $p = 8F$. Then, the interconnect-wire-limited area of a

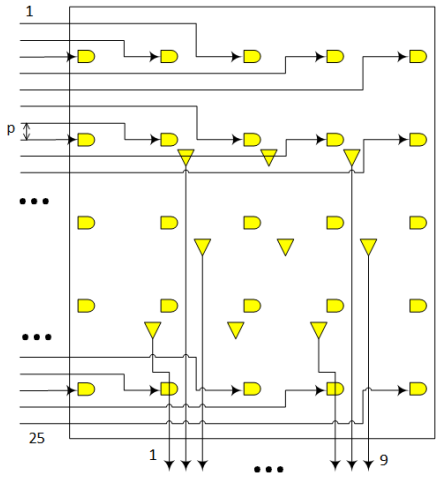

FIGURE 6. Intercore interconnects and neurons for the convolution topology of a neural network. Pitch *p* is marked.

core is

$$a_{wire} = n_{in} n_{out} p^2. \tag{12}$$

The speed of a neural gate is determined by its fan-in, i.e., the number of synaptic operations connected to one output neuron that can be performed in parallel. We will assume the fan-in of digital CMOS to be 2 and all analog devices to be 32. The fan-in for spiking NN is assumed unlimited regardless of a device. We will apply this limitation to the bottom–up benchmarks and consider that it is satisfied for the top–down benchmarks. In the case when the devices forming neurons have a limited fan-in f_i, they need to be cascaded [30]. The number of levels of cascading is (rounded to the next higher integer)

$$l_{cas} = ceil(\log_{fi} s_{neu}). \tag{13}$$

Then, the number of neurons to form a fan-in in a neural gate is

$$n_{cas} = (f_i^{lcas} - 1)/(f_i - 1) \tag{14}$$

so that the number of neurons per core is

$$n_{cor} = n_{cas} n_{out} + n_{in}. \tag{15}$$

In the cross-connect case, the area of a core (performing a stage of a NN) is (provided that n_{in} is larger than s_{neu})

$$a_{cor} = M_{cor} \left(M_{neu} a_{neu} n_{cor} + M_{syn} a_{syn} n_{out} n_{in} \right) \tag{16}$$

In the convolution case

$$a_{cor} = M_{cor} (M_{neu} a_{neu} n_{cor} + M_{syn} a_{syn} n_{out} s_{neu}). \tag{17}$$

We then take the larger of this estimate and the interconnect wire limit a_{wire} to constitute the area of a core a_{st} for the given stage of CoNN. The time and energy for a stage of the computing workload are

$$\tau_{st} = l_{cas} \tau_{syn} + \tau_{neu} \tag{18}$$

$$E_{st} = r_a s_{neu} n_{out} E_{syn} + n_{out} E_{neu}. \tag{19}$$

If the fan-in of the devices is small, which makes its cascading impractical, the synaptic operations can be performed sequentially. We will be using this method for neural accelerators. In this case, the above-mentioned fan-in factors are set to 1, and instead, the time estimate changes to

$$\tau_{st} = s_{neu} \tau_{syn} + \tau_{neu}. \tag{20}$$

In a multistage network, such as in Fig. 5, each stage is using one core. Therefore, the above-mentioned benchmarks need to be multiplied by the number of feature maps in each stage and then summed over all stages to obtain benchmarks for a computing workload

$$a_{CW} = \sum_{st} a_{st} f_{st} \quad \tau_{CW} = \sum_{st} \tau_{st} \; E_{CW} = \sum_{st} E_{st} f_{st}. \tag{21}$$

In the case where the implementation of CoNN is constrained by area, all the cores in Fig. 5 can be replaced by a single core, and all the stage operations can be performed in a sequential ('time multiplexed') manner. Here, we neglect the energy and delay of storing the intermediate results. Then, in this case, the estimates need to change to

$$a_{CW} = \max_{st}(a_{st}) \quad \tau_{CW} = \sum_{st} \tau_{st} f_{st}. \tag{22}$$

This is the case, for example, for all networks using a digital multiplier in a synapse (labeled "MAC"). This treatment is used for all top–down estimates of implemented chips. Then, the power in a computing workload and the throughput of inferences (not to be confused with the synaptic throughput) in units of inferences per second (IPS) per unit area are

$$P_{CW} = E_{CW} / \tau_{CW} \tag{23}$$

$$T_I = 1/(a_{CW} \tau_{CW}). \tag{24}$$

VIII. PROTOTYPE NEUROMORPHIC CHIPS

We will compare the above-mentioned benchmarks with those for prototype chips fabricated and measured by several groups of researchers. In this section, we consider mostly spiking neuromorphic chips [5], [6], [33]. To them, we apply "top–down benchmarking," i.e., calculate the neuron and synapse values from the total number of synapses, the total area, and the known synaptic throughput. A number of such chips have been previously benchmarked [37]. One should keep in mind the difference between the "bottom–up" and "top–down" benchmarks. The former is for idealized circuits and does not include multiple auxiliary circuits necessary for the operation of a chip. The latter is complete real-life chips and contain all the circuit overheads that are hard to quantify. However, we will, sometimes, put these two types of benchmarks side by side for sanity checks and to extract some insights, given the above-mentioned caveats. We assume that 5% of the chip area is occupied by neurons and the rest by synapses. The area per neuron and the area per synapse are thus

$$a_{neu} \approx 0.05 a_{ch}/(c_{ch} n_{cor}), \; a_{syn} \approx 0.95 a_{ch}/(c_{ch} n_{cor} s_{neu}). \tag{25}$$

Publications mostly quote the energy per synaptic event. We approximate the energy per neuron as

$$E_{neu} = E_{syn} r_a s_{neu}. \tag{26}$$

The synaptic throughput, power, and energy per time step are calculated as in Section VI. The input parameters from cited

publications are collated in [30]. In some cases, the input parameters are not available, so they are calculated from the consistency of quoted synaptic throughput with that calculated from equations in Section VI. Then, we use these inputs to obtain the benchmarks for a synapse and a neuron and calculate benchmarks for various computing workloads described in Section VII.

IX. DIGITAL NEURAL ACCELERATORS

There is another type of neural chip being fabricated, which is commonly called neural accelerators [3]. They are based on traditional digital chips and, in this sense, are different from other neuromorphic hardware. Unlike CPU and GPU chips that implement neural network algorithms in software, neural accelerators have dedicated hardware engines to implement neural networks. They are highly optimized for vector-matrix multiplications, which is the core operation in neural algorithms. In this sense, they present a good comparison for digital CMOS neural networks. The input data for them are collected in [30].

Our treatment of them is similar to that in Section VI but adjusted for the nonspiking type. In this case, the clock frequency determines the time step. Performance of these chips is often quoted in MAC/s. A MAC counts as two floating-point operations (FLOP)—multiplication and addition—although different delay and energy are required between the two of them. A significant share of the area of these chips is occupied by the cache, control circuits, and so on. For the chip estimate, we assume that 10% of the neural accelerator area is occupied by neurons and synapses. We use the inputs collected in [30], obtain the benchmarks for a synapse and a neuron, and then calculate the benchmarks for various computing workloads described in Section VII.

X. RESULTS FOR PHYSICAL PERFORMANCE

The most informative view with the benchmarks was provided by the comparison of operation delay and energy. Such energy-delay plots are provided both for synapses (see Fig. 7) and for neurons (see Fig. 8). In many subsequent energy-delay benchmarks, the following technology options are found to be located in close proximity to each other: the group of xCFd, xCOd, xCJd, and xCHd and the group of xCFa, xCOa, xCGa, and xCPa. In other words, these are NNs with digital CMOS neurons (for the first group) or analog CMOS neurons (for the second group) that have very similar designs within each group. The difference within each group is the type of resistive memory comprising a set of binary bits (for the first group) or the analog resistive elements (for the second group). The analysis shows that different kinds of resistive memory produce noticeable but minor differences. In the following plots, for clarity, we will be suppressing the labels, leaving just one label for each group overall. Also, in some plots, we will group results based on synapses doing the MAC operation regardless of the NN type.

One observes that among the four NN types, their neurons have similar ranges of energy. However, on the average, the delay when ordered from fastest to slowest is as follows: ANN, ONN, CeNN, and SNN. Within each type of NN, the networks with both ME neurons and synapses (xEEa) show the lowest energy. This is in line with benchmarks for the Boolean computing [9]. The networks with both

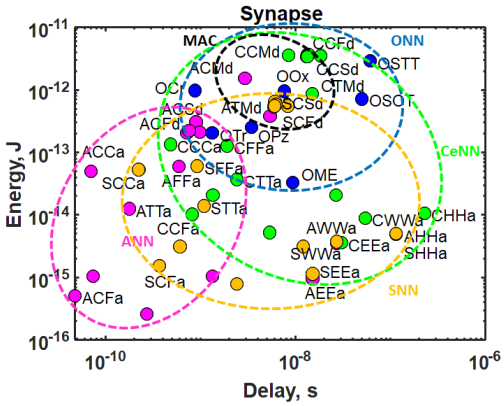

FIGURE 7. Energy versus delay for a synapse in ANN (magenta dots), CeNN (green dots), SNN (gold dots), and ONN (blue dots). Labels for architectures are in [30].

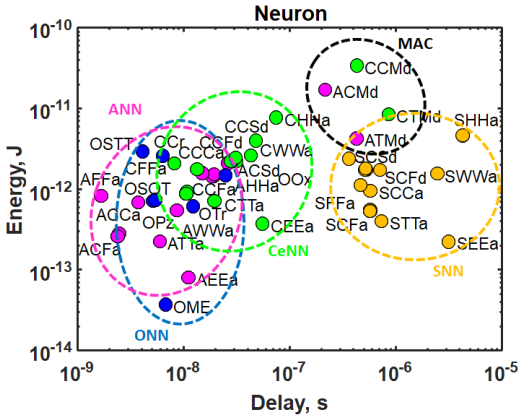

FIGURE 8. Energy versus delay for a neuron in ANN (magenta dots), CeNN (green dots), SNN (gold dots), and ONN (blue dots). Labels for architectures are in [30].

ferroelectric neurons and synapses (FEFET) show the fastest speed. This is a result of the combination of relatively fast switching of a transistor and the assumption that only a single ferroelectric transistor is capable of performing the neuron function. The NNs based on a multiplier and an adder in each synapse (MAC) prove to be the slowest and the most energy consuming due to a large number of switching transistors in each such element. In general, NNs with analog neurons are faster and more energy efficient than similar NNs with digital neurons. This is due to the fact that the neural function is performed in parallel in analog NNs rather than synapse-sequential in digital NNs.

The separation of the four NN types is not so clear for synapses. The overall ranges of energy are similar between the types. However, on the average, the NN delay ordered from fastest to slowest is as follows: ANN, SNN, CeNN, and ONN. The difference from the relationship in neurons is due to the fact that the synapses are similar between ANN and SNN, and the difference arises from the operation on the core level. Other trends for synapses are similar to those for neurons.

The relative relation between energy and delay for a whole computing workload (LeNet in this case, see Fig. 9) closely resembles that for neurons. On the average, ANNs are faster

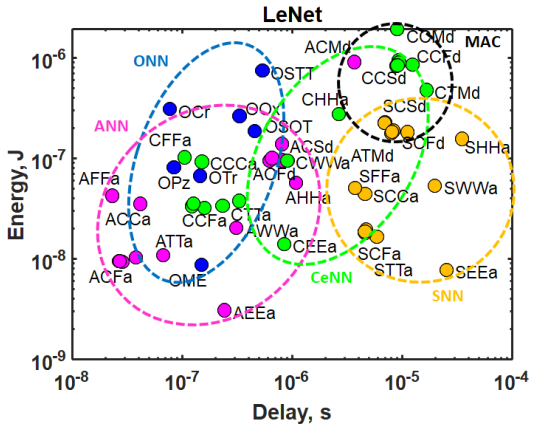

FIGURE 9. **Energy versus delay for one inference in a circuit implementing the LeNet CoNN: ANN (magenta dots), CeNN (green dots), SNN (gold dots), and ONN (blue dots). Labels for architectures are in [30]: bottom–up benchmarks.**

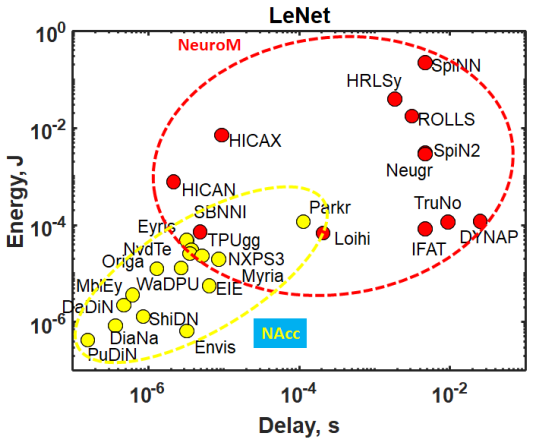

FIGURE 10. **Energy versus delay for one inference for CMOS circuit implementations of the LeNet CoNN with various neural accelerators (yellow dots) and neuromorphic spiking chips (red dots): top–down benchmarks. Data from [30, Tables 3 and 5] are used.**

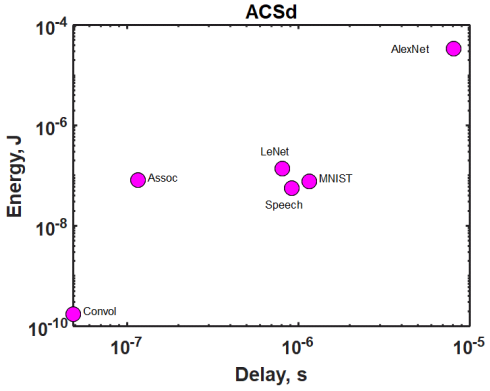

FIGURE 11. **Energy versus delay for one inference in various workloads implemented with digital neurons and SRAM synapses.**

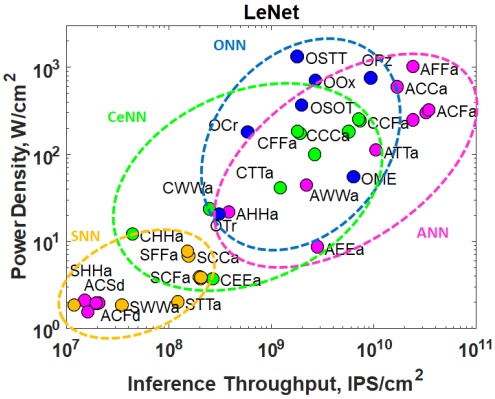

FIGURE 12. **Dissipated power density versus inference operation throughput per unit area in a circuit implementing the LeNet CoNN. Labels for architectures are in [30].**

than ONNs by about a half an order of magnitude, faster than CoNN by an order of magnitude, and faster than SNN by two orders of magnitude. ME devices are more energy efficient than analog neurons by about an order of magnitude. They are more energy efficient than digital neurons by about another order of magnitude. The redeeming quality of SNNs is built-in learning via spike-dependent timing plasticity (SDTP) [4].

Now, we include top–down benchmarks for experimentally demonstrated integrated neuromorphic chips and neural accelerators (see Fig. 10). We notice that the neural accelerators are within an order of magnitude of agreement with the bottom–up benchmarks of their similar technology (ACMd) implementation: $\sim$1 μs and $\sim$100 nJ. Neuromorphic spiking chips, in general, prove to be slower than neural accelerators. The difference depends on whether they are designed to run at biologically feasible firing rates (few tens of Hertz, e.g., ROLLS) or at an accelerated rate (tens of kilohertz, e.g., HICANN). These are still slower than the clock rates of hundreds of megahertz that are used in neural accelerators. Neuromorphic SNN chips have a lower power of operation than neural accelerators. However, their speed (determined by the firing rate) is much slower than that of neural accelerators (determined by the clock rate). As a result, the energy per inference (proportional to the product of the operation delay and power) proves to be higher in neuromorphic chips.

Energy and delay for various workloads (but the same hardware) are shown in Fig. 11. The numerical values are determined by the size of the overall network, mostly the number of MACs in it [30]. The relation between energy and delay between the various types of hardware looks similar from workload to workload.

XI. THROUGHPUT AND DISSIPATED POWER

Circuit performance can be represented as computing throughput plotted versus dissipated power (see Fig. 12). One notices that spintronic networks have a higher per unit area throughput due to the small size of their implementation of neurons and synapses. However, this higher throughput results in very high dissipated power. If we apply a cap on allowed power dissipation (see Fig. 13), some architectures with leading throughput values (e.g., AFFa and ACFa) are scaled back proportionally. In this case, only low-energy spintronic options maintain high throughput (e.g., ATTa).

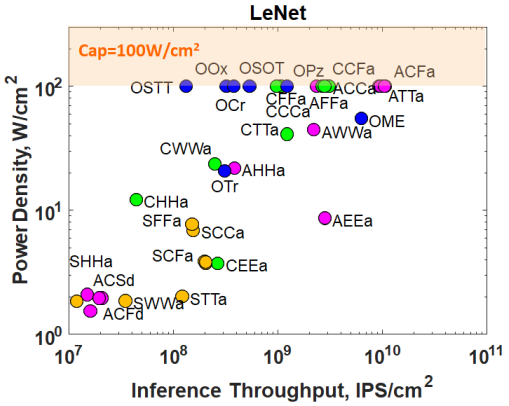

FIGURE 13. Dissipated power density versus inference operation throughput per unit area in a circuit implementing the LeNet CoNN. Power is capped to 100 W/cm^2 and throughput is scaled proportionally.

XII. CONCLUSION

In summary, the developed methodology described in this article enables quantifying the effect of devices and NN types on the performance, power, and area of NNs. ANN and ONN show a higher speed of operation at comparable energy versus CeNN and SNN. This translates into a larger inference throughput, especially under the limitation of power dissipation. The trend is confirmed by a comparison of actual fabricated functional neuromorphic chips (which are SNN) and neural accelerator chips (which are ANN based). Within each NN type, the ones based on multipliers and adders (MAC) prove to be less efficient, while ones based on analog neurons and synapses prove to be more efficient in both speed and energy of operation. Among those, ferroelectric devices show higher speed and spintronic devices (especially based on ME switching) show lower energy of operation. It is important to stress that the conclusions relate to inference and do not cover learning. Also, they relate to one type of NN topology and one class of computing workloads. SNNs are especially amenable to unsupervised learning, and this advantage is not comprehended in the present benchmarks.

ACKNOWLEDGMENT

The authors would like to thank N. Srinivasa, M. Mayberry, S. Manipatruni, G. Chen, R. Krishnamurthy, C. Pan, A. Naeemi, D. Hammerstrom, M. Davies, E. Culurciello, D. Strukov, K. Roy, and W. Porod for their discussions and critique.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Proc. IEEE*, vol. 86, no. 1, pp. 82–85, Jan. 1998.

[2] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Toronto, ON, Canada, vol. 28, Jun. 2017, pp. 1–12.

[3] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

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

[5] S. Furber, "Large-scale neuromorphic computing systems," *J. Neural Eng.*, vol. 13, no. 5, 2016, Art. no. 051001.

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

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[8] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proc. IEEE*, vol. 101, no. 12, pp. 2498–2533, Dec. 2013.

[9] D. E. Nikonov and I. A. Young, "Benchmarking of beyond-CMOS exploratory devices for logic integrated circuits," *IEEE J. Exploratory Solid-State Comput. Devices Circuits*, vol. 1, no. 1, pp. 3–11, Dec. 2015.

[10] D. E. Nikonov and I. A. Young. (2019). *Benchmarking of devices in the Nanoelectronics Research Initiative*. [Online]. Available: https://nanohub.org/tools/nribench/browser/trunk/src

[11] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, nos. 1–3, pp. 239–255, 2010.

[12] C. D. Schuman *et al.*, "A survey of neuromorphic computing and neural networks in hardware," 2017, *arXiv:1705.06963*. [Online]. Available: https://arxiv.org/abs/1705.06963

[13] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: Analog computing," *Proc. IEEE*, vol. 107, no. 1, pp. 108–122, Jan. 2019.

[14] Q. Liu, G. Pineda-García, E. Stromatias, T. Serrano-Gotarredona, and S. B. Furber, "Benchmarking spike-based visual recognition: A dataset and evaluation," *Frontiers Neurosci.*, vol. 10, p. 469, Nov. 2016.

[15] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, Feb. 2018.

[16] S. Agarwal *et al.*, "Achieving ideal accuracies in analog neuromorphic computing using periodic carry," in *Proc. Symp. VLSI Technol.*, Kyoto, Japan, vol. 2017, pp. T174–T175.

[17] A. Canziani, E. Culurciello, and A. Paszke, "Evaluation of neural network architectures for embedded systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Baltimore, MD, USA, May 2017, pp. 1–4.

[18] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016, *arXiv:1605.07678*. [Online]. Available: https://arxiv.org/abs/1605.07678

[19] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," 2018, *arXiv:1812.01739*. [Online]. Available: https://arxiv.org/abs/1812.01739

[20] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in *IEDM Tech. Dig.*, San Francisco, CA, USA, Dec. 2017, pp. 6.1.1–6.1.4.

[21] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, Oct./Nov. 2017, pp. 1916–1920.

[22] M. J. Marinella *et al.*, "Multiscale co-design analysis of energy, latency, area, and accuracy of a ReRAM analog neural training accelerator," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 86–101, Mar. 2018.

[23] C. Pan and A. Naeemi, "Non-Boolean computing benchmarking for beyond-CMOS devices based on cellular neural network," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 2, pp. 36–43, Dec. 2016.

[24] M. S. Zaveri and D. Hammerstrom, "Performance/price estimates for cortex-scale hardware: A design space exploration," *Neural Netw.*, vol. 24, no. 3, pp. 291–304, 2011.

[25] J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers Neurosci.*, vol. 7, p. 118, Sep. 2013.

[26] A. Sengupta, A. Ankit, and K. Roy, "Performance analysis and benchmarking of all-spin spiking neural networks (special session paper)," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, May 2017, pp. 4557–4563.

[27] Z. Du *et al.*, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," in *Proc. 48th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Waikiki, HI, USA, Dec. 2015, pp. 494–507.

[28] 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," *Sci. Rep.*, vol. 6, Jul. 2016, Art. no. 029545.

[29] A. Sengupta, A. Banerjee, and K. Roy, "Hybrid spintronic-CMOS spiking neural network with on-chip learning: Devices, circuits, and systems," *Phys. Rev. Appl.*, vol. 6, Dec. 2016, Art. no. 064003.
[30] D. E. Nikonov and I. A. Young, Supplementary material to this paper, 2019.
[31] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.
[32] T. Sharp, F. Galluppi, A. Rast, and S. Furber, "Power-efficient simulation of detailed cortical microcircuits on SpiNNaker," *J. Neurosci. Methods*, vol. 210, no. 1, pp. 110–118, Sep. 2012.
[33] Y. Le Cun *et al.*, "Handwritten digit recognition: Applications of neural network chips and automatic learning," *IEEE Commun. Mag.*, vol. 27, no. 11, pp. 41–46, Nov. 1989.
[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25. Stateline, NV, USA, Dec. 2012, pp. 1097–1105.
[36] C. S. Thakur *et al.*, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers Neurosci.*, vol. 12, p. 891, Dec. 2018.
[37] B. Chatterjee, P. Panda, S. Maity, A. Biswas, K. Roy, and S. Sen, "Exploiting inherent error resiliency of deep neural networks to achieve extreme energy efficiency through mixed-signal neurons," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 6, pp. 1365–1377, Jun. 2019.