Computing with hardware neurons: spiking or classical? Perspectives of applied Spiking Neural Networks from the hardware side.

Sergei Dytckov
University of Turku
Finland

Masoud Daneshtalab
Royal Institute of Technology
Sweden

***Abstract*—while classical neural networks take a position of a leading method in the machine learning community, spiking neuromorphic systems bring attention and large projects in neuroscience. Spiking neural networks were shown to be able to substitute networks of classical neurons in applied tasks. This work explores recent hardware designs focusing on perspective applications (like convolutional neural networks) for both neuron types from the energy efficiency side to analyse whether there is a possibility for spiking neuromorphic hardware to grow up for a wider use. Our comparison shows that spiking hardware is at least on the same level of energy efficiency or even higher than non-spiking on a level of basic operations. However, on a system level, spiking systems are outmatched and consume much more energy due to inefficient data representation with a long series of spikes. If spike-driven applications, minimizing an amount of spikes, are developed, spiking neural systems may reach the energy efficiency level of classical neural systems. However, in the near future, both type of neuromorphic systems may benefit from emerging memory technologies, minimizing the energy consumption of computation and memory for both neuron types. That would make infrastructure and data transfer energy dominant on the system level. We expect that spiking neurons have some benefits, which would allow achieving better energy results. Still the problem of an amount of spikes will still be the major bottleneck for spiking hardware systems.**

1. Introduction

Neural networks have gained quite a lot of attention last decades. From one side, there are large national and international projects such as Blue Brain, Human Brain project, the BRAIN initiative and a number of separate research groups bring up the advances in spiking neural networks (SNN). From another side, classical neurons with sigmoidal activation function show the possibility to outperform other machine learning algorithms. They are found to be very efficient for text, audio, and image recognition tasks [1]. The Deep Neural Network (DNN) architectures as Convolutional Neural Networks (CNN) and Deep Belief Networks are extensively researched and already widely used in practice.

The main area of research on spiking neurons is in neuroscience (e.g. Brain modelling). Applied architectures for spiking neurons are much less explored, but show promising results. In simple cases, as the classification of the IRIS dataset [2] or prediction [3], SNN show accuracy comparable or even better than classical neurons. Adapting CNN to spiking neurons shows slightly worse accuracy on complex image recognition tasks [4], [5].

Low energy consumption is widely believed to be an inherent feature of SNN. Spiking neuron, implemented using physical characteristics of transistors working in subthreshold regime, show the promising energy result of a few pJ per spike [6]–[10], potentially making neuromorphic hardware (VLSI of silicon spiking neurons) suitable for an energy efficient computing of applications, where accuracy is not critical. When a new platform is introduced, a comparison is typically made with either a general purpose CPU or previous works on the same type of neuron. To our knowledge, there are only two works have compared spiking and non-spiking systems. Another work compares spiking and non-spiking digital VLSI ANN implementations with an application to MNIST database classification [11]. But the reported values are controversial. For energy comparison, 500 cycles of SNN are considered, resulting in more than x3000 times more energy consumption than classical ANN. That is equal to x7 larger energy consumption per cycle for SNN. Although another test on a small dummy network shows that SNN is x2 times more energy efficient per cycle. Despite this, the conclusion is that SNN are not competitive to classical ANN due to a large simulation time although SNN cost twice smaller area. FPGA implementations are explored in [2]. But only area (FPGA slice) utilisation is considered, which is slightly lower for the simple spiking neuron model. In this paper, we explore the hardware realization of spiking neurons and networks from both digital and analog implementation and compare them with classical neural systems. The key objective is in the energy comparison, while exploring the possibility of neuromorphic hardware to be used in practical applications.

This paper is built in the order of growing complexity. Section 2 provides a necessary background on neural networks and the neuron models. Section 3 explains the methods and metrics we used for comparison of spiking and non-spiking neural hardware. Section 4 gives a comparison of individual neurons and their individual operations. Section 5 compares hardware accelerators for ANN on the

system level. Section 6 explores an effect of emerging resistive memory technology on the implementation of ANN hardware.

2. Background on neural networks

On the level of individual processing blocks, a classical neuron receives N pairs of inputs x and weights w and performs N multiplications on them, then the weighted inputs are summed with $N-1$ additions and send to the sigmoidal-like kernel, equations (1,2). The whole operation can be easily pipelined, time multiplexed and partially computed. Sigmoidal kernel is typically implemented as a LookUp Table (LUT).

$$y = F(\sum_{i}^{N} \omega_i x_i) \tag{1}$$

$$F(t) = (\frac{1}{1+e^i}) \tag{2}$$

Spiking neurons are very similar to classical neurons from the architectural point of view. A neuron has a fixed amount of weighted inputs - synapses, and the kernel performs some operations on them. The activation function is replaced with a more complex model, which simulates the behavior of neurons membrane potential in time and its reaction to input - spikes. Spiking neurons receive a distributed in time sequence of spikes. Spikes are essentially boolean values, so the weights are injected (accumulated) directly into the neuron kernel. The computational complexity highly depends on the model. Here we concern the simplest Leaky Integrate and Fire model (LIF) [12], see equation (3). This model describes the membrane potential V, which receives input I and constantly leaks toward resting potential E with the speed controlled by τ.When the threshold potential V_{thr} reached, the membrane potential is reset to V_{reset} and neuron emits an output spike. The LIF model is able to fire only a regular output spiking rate in response on a constant input, which is not enough for biological simulations, but it is fine for practical computations. In the simplest case, input I is just a pulse corresponding to a weight of an input spike, however synapse may have a more complex model, which provides input distributed in time, so called current based synapse.

$$\begin{aligned} &\frac{dV}{dt} = I + (V(t) - E)/\tau \\ &if\ V = V_{thr} \Rightarrow V = V_{reset} \end{aligned} \tag{3}$$

In this paper, we consider ANN as a machine learning method, implementing a practical task. Training of an ANN consists of adjusting synaptic weights. It is typically a very long iterative process. The best known learning rules are error backpropagation for classical neurons [13] and spike-timing dependent plasticity (STDP) for spiking neurons [14]. To make it clear, we do not consider the learning process in this paper.

Implementation of both spiking and classical neurons is explored in both digital and analog domains. For classical neurons, the benefit of analog circuits is significant [15], [16], but we were unable to find any recent works with the breakdown data of individual components to include in this paper. The behavior of a spiking neuron is modeled efficiently by a few transistors working in the deep subthreshold regime [6]–[10], plus it inherently supports the continuous asynchronous simulation. Digital spiking neurons suffer from a complex model, which requires continuous recalculation even when there is no input spike. This becomes worse when the input data should be encoded by spike-rate which increases simulation time significantly. To address this issue, over-simplified models are commonly used.

Large classical ANN may be easily simulated on a limited number of time-multiplexed neurons (either digital or analog) as there is no state in classical neurons. Digital spiking neurons may be time-multiplexed, but it is costlier, as neuron state (membrane potential V in the simplest case of equation (3)) need to be saved. Analog spiking neurons, theoretically, may also be time-multiplexed, if there are no cyclic connections in a network, but, to our knowledge, this has not been explored in the literature. Thus, most of the digital hardware designs for both neuron designs exploit the time-multiplexing feature to minimise area consumption, while analog spiking hardware is typically implemented in a full parallel way.

3. Methods of comparison

To have a fair comparison, we considered two granularity levels: a) the level of individual neurons and b) the whole system level. The key assumptions to make a comparison possible are as following: (1) one input of a classical neuron is equivalent to one synaptic connection of a spiking neuron and (2) the structure of neural networks (the number of neurons and their connections) for both classical and spike neuron types are similar for the same applications.

Energy per spike is measured as a key metric for analyzing analog neurons, however this metric depends on a benchmark. For spiking neurons, most commonly, the energy consumption is recorded by applying a strong constant input, which charges the capacitor above the threshold of a neuron. This benchmark is not representative, as in practice a neuron integrates multiple input spikes, distributed in a significant time window, to produce an output spike. Thus, the total amount of input, integrated into the neuron capacitor, commonly exceeds a threshold several times, due to the leakage component. Current injection infrastructure, as digital-to-analog converter (DAC), needed to convert synaptic weighted into input current, is rarely included in a benchmark, making the estimations too optimistic. We find that the *energy per synaptic event* (a single input spike coming to a single neuron) is more representative and realistic metric. It is rarely used on the level of individual neurons, but may be extracted from the full system designs.

For classical neurons, power and timing characteristics are commonly available. Out of that, we estimate energy consumption per operation. In fact, we calculate the energy

consumption per single input to classical neurons and compare it with a single input or output spike.

For the full system comparison, in most cases, we also estimate energy consumption using power values and timing. We try to scale the energy values for a single benchmark, which is the edge detection on a small 18x18 image with 3x3 kernel size. That is the first and the main comparison metric in this paper. Edge detection is a simple CNN, consisting of a single convolutional layer of 4 directional convolutions and one max, sum or average pooling layer. Classical neurons are able to represent both positive and negative outputs, thus only 2 convolutional maps, with an absolute function, may be used.

In some cases, [17]–[20], it is hard to scale the reported energy values to get the first metric due to a complex benchmark used in the original works. Thus, we take the second metric, which is the number of *giga-operations per second per watt* ($GOPS/W$) for classical neurons or *giga-synaptic operation per second per watt* (GSOPS/W) for spiking neurons. With the assumption (1), we may directly compare $GOPS/W$ with $GSOPS/W$. This metric shows the system energy efficiency level, however, it does not take into account the SNN application specific that more than one spike is used for a single operation.

The largest part of the systems considered in this work are oriented towards CNN as the state of the art ANN architecture. The results, presented in Sections 4 and 5, should be taken with a grain of salt, as a simple linear scaling applied in this work is not accurate. For example, in Origami system [21], reported energy and $GOPS/W$ were obtained from the configuration of 7x7 kernel size with 4 hardware neurons implementing 8 convolutional kernels in parallel. The values were scaled down x22 times to represent a convolution window of 3x3 pixels with just 2 kernels.

4. Silicon neurons

Table 1 combines the data for the classical neuron cores. Values for individual synapses (basically multiplier, adder and an insignificant fraction of an activation function) are calculated from the system level data. Energy values are not reported directly in the original published works, but we have extracted them from some other reported data (like power and timing). These extracted data are highlighted with an underline in all tables throughout the paper.

In [22], a neuron with 16x 16-bit inputs was implemented in a pipeline manner. Parallel multipliers in the first layer, adder tree in the second, and finally sigmoidal kernel implemented as a piecewise linear approximation with LUT. In the original paper a power of 16 neurons is reported, thus we are able to extract the energy for a single neuron, which is $0.5pJ$ per input. This value should be taken accurately, as the system was benchmarked with full CNN applications, most of which had a small convolutional kernel and includes pooling layers (e.g. average operation on a 2x2 kernel). Thus, the neurons were highly underutilised and consumes less power. In [23], a defect tolerance of hardware neurons is explored. A single layer of 10 neurons, each with 90x 16-bit inputs, is implemented. Dividing the total energy by the number of synapses, we calculate the energy consumption per synapse to be $78pJ$. For the last row of Table 1, we implemented a single neuron with 16-bit fixed point input and synthesized it by Cadence RTL tool with the $65nm$ process.

TABLE 1. CLASSICAL DIGITAL NEURON CORE COSTS AND EFFICIENCY

Ref.	Tech. node, nm	Area, μm^2	Power, W	Energy/syn., pJ
[22]	65	<u>52903</u>		<u>0.5</u>
[23]	90			<u>78</u>
Our exp.	65	18643	7.2e-03	<u>8</u>

Data on spiking neurons are quite fragmented. We have collected the data from different papers in Table 2. Most of the digital designs have been targeted field-programmable gate array (FPGA). As the power value of FPGA is only reported for the whole chip, no matter how much of it is utilised, their reports were not useful for us. A review of the large amount of models, implemented as analog circuits, has been presented in [6]. Most of the models there are biologically plausible, which makes them energy and area costly. We focus on simple models, which show better area and energy results, and potential for practical applications.

In [7], different modulation methods with current injecting into membrane capacitor are presented. The most efficient is the current modulation, which consumes $0.25pJ$ with 7-bit current precision. This energy value is recorded when a maximal input, higher than the threshold, is injected. The large area of the neuron is driven by the need for a large DAC to implement multiple current levels. The energy consumption in the realistic case is likely to be larger due to a clocked latch comparator, which dissipates a significant portion of the neuron power in case of continuous simulation.

In [8], both analog and digital designs are compared at $65nm$ process. In both cases, an input spike is injected with the time-based method as following. In the digital design, the input is integrated into a counter based on a number of cycles, similarly, in the analog design, the number of equal current pulses is integrated into a capacitor. The analog design consumes $2pJ$, while digital design $41pJ$. That is the difference in an order of magnitude. Another result for a digital neuron is obtained from the system level values of TrueNorth architecture [24]. See the details of the architecture in Section 5. The reported energy consumption is $26pJ$ per synaptic event. This value, however highly depends on the network connectivity and activity.

In [9], a model capable of a wide variety of biologically plausible responses is implemented as an analog circuit. A strong input voltage causing $100Hz$ output spike frequency results in 8.5-$9pJ$ energy consumption per output spike at 0.35 m process. However, no spike injection circuitry is presented in that work. In [10], a neuron together with a synapse, implementing STDP rule, were implemented at $90nm$ process. A neuron core was studied separately with an

external voltage source, i.e. without any injection circuitry. The area for the neuron core and synapse is 442 μm^2 and 4823 μm^2 respectively, while each synapse in this model should be physically implemented as it contains a weight value internally in a capacitor. The resulting energy consumption of the neuron core is just $0.4pJ$. The plasticity rule is activated on both input and output spikes and consumes $0.37pJ$ per each spike, but in this case energy dissipation for charging the neuron cell capacitor is omitted. We ignore plasticity rules in this work, but, we include it in our table due to the energy efficiency level of the neuron. In [25], a model with biologically realistic spike behavior was implemented in silicon. It occupies a significantly larger area than other models, even though the $90nm$ technology was utilised. The reported energy consumption per spike is $1pJ$, and the power consumption in the resting state is reported being 35 times less than in the spiking mode. Current injection circuitry is omitted.

In [18], [20], a chip for spiking convolution is implemented, see the details of the architecture in Section 5. There are two unique features of this implementation. The first is the presence of positive and negative spike types and two comparators to detect two thresholds. The second feature is the calibration circuits to address hardware mismatch. This requires 2930 μm^2 area at $0.35\mu m$ technology. The energy consumption on the chip level highly depends on the convolutional kernel size. To minimize the overhead of the infrastructure (I/O pads, memory), we assume the situation with the highest available kernel size when each input spike is addressed to every neuron. The energy consumption in this case is $4pJ$ per synaptic event.

TABLE 2. SPIKING NEURON CORE COST AND EFFICIENCY

Ref.	Tech. node, nm		Area, μm^2	Energy/ spike, pJ	Energy/ syn., pJ
[7]	65	Analog	5000	0.25	
[8]	65		1300	2	
[9]	350		1600	8.5-9	
[10]	90		442 / 4823	0.4	0.37
[25]	90		2980	1	
[20] [18]	350		2930		4
[8]	65	Digi.	538	43	
[24]	28				26

4.1. Discussion

The first thing to note from the results is that digital spiking neurons are not energy efficient compared with the analog alternative, despite having the benefit of technology scaling. Analog spiking designs, in turn, have about the same or better energy efficiency than classical neurons. However, in the majority of applications, the spike rate encoding is used, which requires to process a large spike train to extract an output. This mitigates the advantage of spiking neurons. Moreover, the values in Table 2 are optimistic due omitting injection circuits in most of the silicon spiking designs.

Although values in Table 1 and 2 have a large variance and there are not enough data on classical neurons, classical digital neurons consume in a range of tens of pJ per one synapse activation while spiking neurons consume in range of single pJ. That would be our initial point of view on the problem: analog spiking neurons consume about an order of magnitude lower energy per single operation (synapse / spike).

5. Silicon neural network system

Table 3 has an overview of full system designs for both classical and spiking neurons. We focus on CNN as the state of the art ANN architecture. We select the hardware architectures reporting the energy values possible to scale towards the same benchmark - the edge detection on 18x18 image with 3x3 convolution kernel and 4 directional maps. To compare the energy efficiency of systems with different types of neuron, we introduce the energy column, by estimating the time required to process 18x18 image and multiplying it by reported power value or scaling the reported energy results where available. All the results extracted from values reported in the original papers are highlighted with the underline.

Due to limited available materials in hardware realization of spike-based systems, in order to complement the results, we have designed our baseline spiking neural platform and performed the simulation with the selected benchmark to extract the energy values. We used the analog circuit of spiking neuron [7], as the power of separate components of the neuron is reported. Neurons are placed into a cluster (56 neurons in a cluster) and connected through a regular 2D mesh with a packet-switch dimension-order routing. Each cluster has a limited number of input axons. Spikes are delivered as a unicast packets through the network based on a destination address and multicasted inside the cluster based on the local axon ID. The design is modeled with systemC (based on Noxim simulator [28]) and the energy consumption is calculated based on the system activity as network traffic, number of memory accesses, number of spikes generated... The energy values of all the cluster components in each cluster have been back annotated from the synthesized results extracted with Cadence RTL tool, network on chip energy have been back annotated with the help of Orion simulator [29].

TABLE 3. FULL SYSTEM COMPARISON

Ref.	Tech. node, nm			Benchmark energy, J	GOPS/W
[22]	65	Classical	Digital	1e-08	931
[26]	28			1e-08	412
[17]	45				230
[21]	65			2e-08	369
[27]	65	Spiking	Analog	1e-07 – 3e-06	
Our exp.	65			1e-07 – 5e-06	
[20] [18]	350				1.4–225
[19]	350		Digi.		2.6-7.5
[24]	28			5e-06	46-400

5.1. Discussion

From Table 3, we can see that the energy consumption on the selected benchmark for systems using non-spiking neurons is on the level of tens of nanojoules. The energy for spike-based systems is larger for about 1-2 orders of magnitude, despite the fact that spiking neurons individually are more energy efficient , see Section 4. This is mainly due to the use of spike-rate encoding, i.e. to encode an 8-bit number a large spike sequence is required. Together with that, power hungry components such as current injection circuits, weight storage and interconnection network also play the dominant roles in the system energy. In our experiment, described in Section 5.2, the impact on the *energy per spike* of infrastructure for neurons (interconnection network, memory storage, injection circuits) imposes almost two orders of magnitude energy overhead, from $0.25pJ$ in [7] to $13pJ$ in our experiment. The results in [27], where the interconnection network is simplified almost to a level of direct connections, 98% of total energy is consumed by memory accesses. In classical systems, infrastructure also consumes the most energy although the impact of the neuron itself is also significant, e.g. in the DianNao architecture [22], the arithmetic block consumes about 27% of the total energy.

It is hard to extract energy values from neuromorphic systems [18]–[20] as the power dissipation is reported on the chip level and benchmark is not clearly defined, so that we use the second available metric - $GOPS/W$. In the Caviar systems convolutional chip [18], [20], the kernel size may vary in the range from a single destination up to 32x32 window. When each packet with a spike coming to the chip is delivered to a single neuron, all the I/O costs is paid for a single activation. This corresponds to a low $GOPS/W$ value. When each spike is delivered to multiple destination neurons the infrastructure energy consumption is amortised by a large amount of synaptic operations. The higher value of 225 GOPS/W in Table 3 for [18], [20] corresponds to the maximum kernel size. We consider this high value for comparison as it represents the situation when hardware resources are optimised for a specific task, as it is done in [21], [26], where the amount of multipliers corresponds to the size of convolutional window. Digital systems approach [19] shows much worse energy efficiency of $7.5GOPS/W$, which only slightly depends on kernel size. This shows that the energy consumption of digital neuron simulation is comparable to infrastructure energy. The TrueNorth architecture [24] also reports a high energy efficiency up to $400GOPS/W$ for a high-connection network. Classical neural systems report an energy efficiency from 230 to $930GOPS/W$, which is achieved by high data reuse and data transfer minimisation as a sequence. These values show that both spiking and non-spiking systems are on a similar level of energy efficiency *per synapse*.

The main conclusion is that the hardware efficiency of neuromorphic platforms is about the same as for classical neuron systems, in terms of single operations. However, the single operator (spike) is typically not enough to represent data, as spike-frequency encoding is commonly used, resulting in significantly higher energy consumption. Thus, a primary milestone to enable neuromorphic computing for applied applications would be the development of applications implementation with alternative (time-based) data encoding requiring less number of spikes. This would allow breaking the distance between spiking and classical systems. Such techniques are known for quite a long time [30], but they are rarely utilised.

From another side, even if we assume that spiking applications with single spike data encoding would be developed, resulting in a similar energy consumption of spiking and classical ANN, the area overhead of classical systems is much lower due to time-multiplexing designs, whereas spiking systems have to be implemented in parallel. Despite the size of a silicon spiking neuron measured in thousands of square micrometers, millions of neurons are required to perform, for example, an image recognition with an acceptable resolution. Thus, we do not expect spiking neural systems to be able to compete with classical approach in computational applications.

6. Memristor crossbar based systems

The emergence of resistive memory cells opens up a new horizon in energy efficiency for ANN. Memristors, combined in a crossbar fashion - memristor crossbar (MC), replace SRAM memory for weight storage and implement a vector-matrix multiplication of input voltages to memristor conductances. That is beneficial for spiking neurons, where SRAM access energy is dominant (98% in [27]), and for classical neurons, where both memory and synaptic multiplications are important (66% in [22] Together with energy, MC is highly dense and scalable, which is important for densely connected ANN, where synapses occupy a huge area in VLSI circuits.

Sneak paths are a problem with a pure MC, as current may go through any path of it, even though we want to access a specific cell. Classical ANN allows mitigating this, as all the inputs are injected simultaneously during the read operation [31], [32]. SNN do not have this feature and thus need some workaround to address the problem of sneak paths, as well as memristor programming requires an access to individual elements. Some works utilise 1T1M (one transistor per memristor) methodology, which is used to access individual memristors and remove the sneak path problem [33]–[35]. Another method of mitigating sneak paths during write operation is to apply half of programming voltage to all the rows and columns except of a selected one, which receives a whole programming voltage on its row and zero voltage on its column [32], [36].

A property of memristors to tune resistance gradually, when small programming pulses are applied, is very well popular in neuromorphic community. It allows emulating STDP learning rule right in the neurons without much extra circuitry. In this paper, we consider only a forward path of ANN, assuming that ANN is pre-trained offline. This is an ideal scenario, yet it is quite realistic, for example in sensor

nodes ANN should execute a pre-processing function on input data. Thus we have not considered those works using learning rules such as back propagation for classical and STDP for spiking neurons.

MC, implementing classical ANN, makes an idea of neural approximated computing very attractive (when a tightly coupled neural accelerator approximates parts of an application, see [15], [16]). This idea is explored in [31], [32], where a group of memristive neurons implement two layered ANN serving as universal approximator provides up to $568GOPS/W$ energy efficiency. In this case, one approximated operation (CPU instruction) is used as a metric, rather than one synaptic MAC operation in Table 4. In [31], 270x energy efficiency improvement over Intel Xeon processor and 12x improvement over FPGA for calculating Gaussian distance (part of HMAX benchmark) are reported.

TABLE 4. MEMRISTOR CROSSBAR BASED ARCHITECTURES

Ref.	Tech. node, nm		Energy/syn., pJ
[34]	45	Class.	1.4
[36]	45		1.8–21
[37]		Spiking	0.5
[33]	130		12
[38]	10		41

6.1. Discussion

Emerging resistive memory technology benefits both type of neurons. MC makes memristor access as the basic synaptic operation. This should results in the energy consumption of a few fJ for both neuron types, as shown in Table 4. Although, on this early stage of memristive technologies, the variance in the reported data is very high. With such a low value for synapses, the total system energy consumption would be dominated by other parameters such as data transmission, ADC/DAC (if we assume that communication, especially in large-scale systems is digital). Spike based systems may have some benefits in that. First, as a spike is an event, a minimal network payload is generated, whereas classical neurons exchange N-bit values. Second, data input to MC requires just a single input voltage level, while classical neurons would require multiple voltage levels and more complex DAC, as well as ADC, to read an output. Such, in [39] ADC/DAC is estimated to consume more than 85% of total power and area for 8-bit accuracy. Authors propose to expand MC and use separate rows and columns for each input and output bit, which saves, by their estimations, 60–85% of total power and 50–85% of total area (as MC is very compact). Although another work [34] reports less than 25% of total energy consumption for current source. Third, there are some potential applications (like DVS based surveillance), where input is essentially spiking and dont need a conversion into spikes. DVS, also, typically require a small amount of spikes to represent the data. For example, in [18], sensor registering the silhouettes of two walking people on 128x128 resolution produce 980 spikes at 40 ms, a normal 25 fps camera would produce 16384 input pixels. Thus the amount of data to process is lower for an order of magnitude for DVS. Together with spike-time based processing it would give an order of magnitude of energy efficiency.

7. Conclusion

With currently available technologies, spiking neuromorphic systems lose classical non-spiking systems on every position (accuracy, area, energy, speed). There is an essential disadvantage in spike-rate encoding, which requires multiple spikes to be processed. Having applications with single-spike data representation may bring spike based systems in-line with classical in the energy efficiency. Memristive memory technology gives a huge benefit for both neuron types, replacing the synaptic circuits. With that, synapse and neuron access become an insignificant part of the system energy consumption. Architecturally, classical ANN accelerators would become very similar to spiking designs. With the similar energy consumption in the synapses for both neuron types, other parameters would become the energy bottleneck. We expect that spiking neurons may have some benefits on a system level, which could bring better energy results. Although there is still the need to mitigate an application level inefficiency of spike-data encoding for SNN.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[2] S. Johnston, G. Prasad, L. Maguire, and M. McGinnity, *Artificial Neural Networks: Biological Inspirations*. Heidelberg, Germany: Springer, 2005, ch. Comparative Investigation into Classical and Spiking Neuron Implementations on FPGAs, pp. 269–274.

[3] C. Johnson, G. K. Venayagamoorthy, and P. Mitra, "Comparison of a spiking neural network and an mlp for robust identification of generator dynamics in a multimachine power system," *Neural Networks*, vol. 22, no. 56, pp. 833 – 841, 2009.

[4] 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, 2014.

[5] P. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, July 2015, pp. 1–8.

[6] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Hfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. SAGHI, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, no. 73, 2011.

[7] N. D. B. Phong, M. Daneshtalab, S. Dytckov, J. Plosila, and H. Tenhunen, "Silicon synapse designs for vlsi neuromorphic platform," in *NORCHIP, 2014*, Oct 2014, pp. 1–6.

[8] A. Joubert, B. Belhadj, O. Temam, and R. Hliot, "Hardware spiking neurons design: Analog or digital?" in *IJCNN*. IEEE, 2012, pp. 1–5.

[9] J. H. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour," *Neural Networks*, vol. 21, no. 23, pp. 524–534, 2008.

[10] J. Cruz-Albrecht, M. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and stdp integrated circuits," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 6, no. 3, pp. 246–256, June 2012.

[11] Z. Du, D. D. Ben-Dayan Rubin, Y. Chen, L. He, T. Chen, L. Zhang, C. Wu, and O. Temam, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," in *Proceedings of the 48th International Symposium on Microarchitecture*, ser. MICRO-48, 2015, pp. 494–507.

[12] W. Gerstner and W. Kistler, *Spiking Neuron Models: An Introduction*. New York, NY, USA: Cambridge University Press, 2002.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds., 1988, ch. Learning Representations by Back-propagating Errors, pp. 696–699.

[14] J. Sjstrm and W. Gerstner, "Spike-timing dependent plasticity," *Scholarpedia*, vol. 5, no. 2, Feb 2010.

[15] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-45, 2012, pp. 449–460.

[16] R. St.Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger, "General-purpose code acceleration with limited-precision analog computation," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, June 2014, pp. 505–516.

[17] P.-H. Pham, D. Jelaca, C. Farabet, B. Martini, Y. LeCun, and E. Culurciello, "Neuflow: Dataflow vision processing system-on-a-chip," in *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*, Aug 2012, pp. 1044–1047.

[18] L. Camunas-Mesa, C. Zamarreno-Ramos, A. Linares-Barranco, A. Acosta-Jimenez, T. Serrano-Gotarredona, and B. Linares-Barranco, "An event-driven multi-kernel convolution processor module for event-driven vision sensors," *Solid-State Circuits, IEEE Journal of*, vol. 47, no. 2, pp. 504–517, Feb 2012.

[19] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, no. 12, pp. 2548–2566, Dec 2006.

[20] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Ballcels, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "Caviar: A 45k neuron, 5m synapse, 12g connects/s aer hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *Neural Networks, IEEE Transactions on*, vol. 20, no. 9, pp. 1417–1438, Sept 2009.

[21] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origami: A convolutional network accelerator," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, ser. GLSVLSI '15, 2015, pp. 199–204.

[22] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '14, 2014, pp. 269–284.

[23] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," *SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 356–367, Jun. 2012.

[24] P. Merolla, J. Arthur, R. Alvarez-Icaza, A. Cassidy, J. Sawada, F. Akopyan, B. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. Esser, R. Appuswamy, B. Taba, A. Amir, M. Flickner, W. Risk, R. Manohar, and D. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, pp. 668–673, August 2014.

[25] V. Rangan, A. Ghosh, V. Aparin, and G. Cauwenberghs, "A subthreshold avlsi implementation of the izhikevich simple neuron model," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, Aug 2010, pp. 4164–4167.

[26] F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15, 2015, pp. 683–688.

[27] B. Belhadj, A. Joubert, Z. Li, R. Héliot, and O. Temam, "Continuous real-world inputs can open up alternative accelerator designs," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13, 2013, pp. 1–12.

[28] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*, July 2015, pp. 162–163.

[29] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, 2002, pp. 294–305.

[30] "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 67, pp. 715 – 725, 2001.

[31] B. Li, Y. Shan, M. Hu, Y. Wang, Y. Chen, and H. Yang, "Memristor-based approximated computation," in *Proc. of the 2013 Int. Symp. on Low Power Electronics and Design*, ser. ISLPED '13, 2013, pp. 242–247.

[32] B. Li, P. Gu, Y. Shan, Y. Wang, Y. Chen, and H. Yang, "Rram-based analog approximate computing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, no. 12, pp. 1905–1917, Dec 2015.

[33] C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, Y. Chen, H. Li, Q. Wu, and H. Jiang, "A spiking neuromorphic design with resistive crossbar," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15, 2015, pp. 14:1–14:6.

[34] D. Fan, Y. Shim, A. Raghunathan, and K. Roy, "Stt-snn: A spin-transfer-torque based soft-limiting non-linear neuron for low-power artificial neural networks," *Nanotechnology, IEEE Transactions on*, vol. 14, no. 6, pp. 1013–1023, Nov 2015.

[35] C. Yakopcic, R. Hasan, and T. Taha, "Memristor based neuromorphic circuit for ex-situ training of multi-layer neural network algorithms," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, July 2015, pp. 1–7.

[36] C. Yakopcic and T. Taha, "Energy efficient perceptron pattern recognition using segmented memristor crossbar arrays," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, Aug 2013, pp. 1–8.

[37] T. Tang, L. Xia, B. Li, R. Luo, Y. Chen, Y. Wang, and H. Yang, "Spiking neural network with rram: Can we use it for real-world application?" in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15, 2015, pp. 860–865.

[38] B. Rajendran, Y. Liu, J. sun Seo, K. Gopalakrishnan, L. Chang, D. Friedman, and M. Ritter, "Specifications of nanoscale devices and circuits for neuromorphic computational systems," *Electron Devices, IEEE Transactions on*, vol. 60, no. 1, pp. 246–253, Jan 2013.

[39] B. Li, L. Xia, P. Gu, Y. Wang, and H. Yang, "Merging the interface: Power, area and accuracy co-optimization for rram crossbar-based mixed-signal computing system," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, June 2015, pp. 1–6.