Memory Organization and Structures for On-Chip Learning in Spiking Neural Networks

(Invited Paper)

Clemens JS Schaefer and Siddharth Joshi
Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA
Email: {cschaef6, sjoshi2}@nd.edu

***Abstract*—Spiking Neural Networks (SNNs) offer a promising avenue for developing adaptive intelligence that can operate effectively, when subject to energy constraints. This is especially true for emerging gradient-based SNNs that are designed for continuous learning. However, hardware efficiency and complex SNN functionality are at odds with each other, leading to large design space, with many possible trade-offs. This paper discusses the co-optimization of SNNs and hardware architectures. We compare digital designs and emerging compute-in-memory architectures, in terms of operation. We discuss the effect of different layer-types on memory organization and outline how different memory organization schemes are impacted by the SNN layer-type. Finally, we outline some open problems that must be solved to design the next-generation of intelligent hardware systems.**

I. Introduction

Advances in machine learning (ML) hint at the great potential of biologically inspired, intelligent systems [1]. However, modern ML algorithms are energetically costly. Thus, often, ML systems are much more resource intensive than biological systems in terms of a) peak and sustained power, b) volume and space, and c) computational resources such as number of neurons and parameters. Biological systems perform complex cognitive task with a few thousand neurons and comparatively low energy and power. Thus, spiking neural networks (SNNs) —which operate with greater fidelity to biological systems, are attractive algorithms to endow adaptability to systems operate under energy constraints [2], [3].

SNNs implemented on dedicated hardware architectures have been used to solve a diverse set of problems including LASSO based sparse recovery [4], vision based multi-object detection and classification [5], communication signal and navigation in mobile robots [6]. In particular, SNNs excel at tasks that depend on temporal sequences. For a more general review of SNNs, we refer the readers to other articles that cover neuromorphic sensors [7], spiking neuron circuits [8].

Generally, energy-efficient operation requires joint-design of the hardware and the algorithm [9]. For artifical neural networks (ANNs), this entails carefully considering the trade-offs between ANN accuracy and memory footprint/computational energy/latency. However, it is not immediately obvious, how optimizations that benefit ANNs might benefit SNNs. As an example, consider activation quantization in ANNs [10]. Such quantization can benefit the underlying hardware in two ways: a) lower bitwidth computation and b) lower-bandwidth in the on-chip network. Implementing such quantization in SNNs would not yield the same benefits, since the on-chip networks would still be transmitting spike-events. In fact, lower bitwidth in the neuron computations might yield higher activity rates, which in turn would dramatically increase the energy expended in the NoC. Consequently, energy-efficient hardware design for SNNs must take into account many factors that differ between SNNs and conventional ANNs.

This article presents an overview of recent advances in gradient-based SNNs, followed by a brief overview of two possible families of hardware architectures: a) emerging compute-in-memory (CIM) architectures and b) conventional von Neumann digital architectures. The remainder of this article discusses co-optimization opportunities that open up with these algorithms, as well as highlighting operating regions that favor the different architectures.

II. Spiking Neural Networks

SNNs encode their activity in *spikes*, which encode the time at which a neuron's membrane potential exceeds its threshold. After generation, a spike propagates along an axon onto its destinations, where the issued spike increases or decreases the membrane potential of the destination neuron by a small amount, proportional to the synaptic strength. This simplified model of a neuron's activity often suffices for many tasks [11].

One of the simplest models of a spiking neuron assumes linear charge accumulation on the neuron membrane, with a fraction of the charge constantly leaking towards the *resting potential*. This is the leaky integrate-and-fire (LIF) below:

$$\tau \frac{d}{dt} U = -U(t) + RI(t), \tag{1}$$

The membrane potential at time t is given by $U(t)$ with its time constant τ, and input current $I(t)$ [11]. Spikes ($S(t)$) are generated as, $S(t) = \Theta(U(t) - \vartheta)$ where Θ is the step function and ϑ is the the firing threshold. The LIF model has been widely adopted because it displays some temporal dynamics while remaining analytically tractable. The spike response model (SRM) [11], builds upon the LIF by including additional temporal dynamics. A multilayer SNN composed of discrete-time SRMs can be described by:

$$U_i^{(l)}[n] = \sum_j W_{ij}^{(l)} P_j[n] - \delta R_i[n]. \tag{2}$$

The membrane potential of neuron i in layer l at discrete time-step n is given by $U_i^{(l)}[n]$. W_{ij} represents the synaptic weight between pre-synaptic neuron j and post-synaptic neuron i. Additional time-dynamics are given by:

$$\begin{aligned} Q_j[n+1] &= \alpha Q_j[n] + S_j^{(l-1)}[n], \\ P_j[n+1] &= \beta P_j[n] + Q_j[n], \\ R_i[n+1] &= \gamma R_i[n] + S_i^{(l)}[n]. \end{aligned}$$

where α, β, and γ are time constants. The variable $Q_j[n]$ represents the eligibility trace of the current-based synapse, $P_j[n]$ represents the membrane trace corresponding to $Q_j[n]$, and $R_i[n]$, the reset magnitude.

Recent SNNs have used the SRM models to implement gradient-based learning [12]. Generally, the discontinuity caused by the threshold function ($\Theta(.)$), prevents SNNs from using gradient-based methods. However, recent advances [12], [13] operate on continuous relaxations of the gradient which remain unaffected by discontinuity. An additional modification uses a local, fixed, random classifier, to generate the error terms required to implement weight updates [14]. The membrane potential of the SNN is processed through a sigmoid for the local classifier resulting in a surrogate gradient for training based on the derivative of the sigmoid function. For this algorithm, the pre-synaptic potential ($P_j[n]$), the membrane potential, ($U_i^l[n]$), together with the external error ($E_i[n]$) are required to implement weight updates. This algorithm is well suited to hardware implementations due to its high-performance with limited access to non-local information.

III. Hardware Platforms for SNNs

Hardware platforms designed for SNNs generally consist of multiple neurosynaptic cores (NSC) and some means to communicate between such cores. The NSC, in turn, contains: *i)* a neuron subsystem which implements spike accumulation and all calculations associated with membrane potential dynamics; *ii)* a synapse subsystem which stores associated weights and implements synaptic dynamics; and *iii)* if learning is supported, then additional circuitry to implement synaptic plasticity. Since these systems must accommodate a large number of neurons and synapses, inter-core communication has typically been implemented through digital means, often using address-event-representation for spike communication [15]. The computations associated neural and synaptic dynamics and learning have are generally implemented either in an externally-digital-internally-analog fashion [8], or entirely through digital circuits [4]. The memory subsystem, responsible for storing synpatic parameters, forms a major point of focus for optimizations in these architectures.

A. In-Memory Computing

Since synapses outnumber neurons by multiple orders of magnitude, emerging CIM architectures are an attractive choice for implementing ML [17] and SNN algorithms. These architectures also benefit from recent advances in emerging memory devices that store values in device conductances [16], enabling parallel matrix-vector multiplication calculations within the memory array. One such prototype

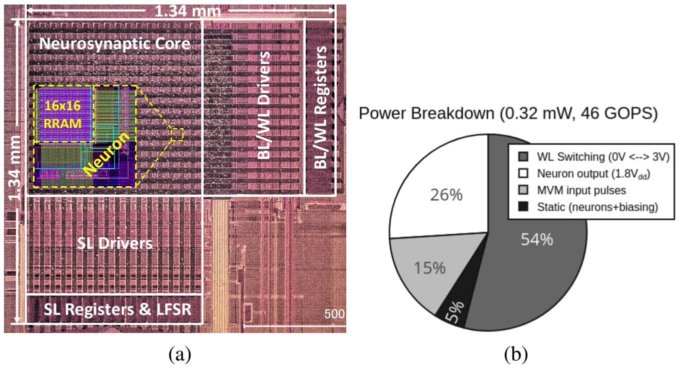

Fig. 1. (a) Microphotograph of a 130 nm CMOS-RRAM compute in memory array composed of multiple interconnected 16×16 sub-arrays each occupying 1849 μm^2 each [16]. (b) The ASIC delivers 74 TMACS/W (144 TOPS/W) computational efficiency, with 46 GOPs peak MVM performance while consuming just 300 μW power.

is shown in 1. This, brain-like, merging of computing and memory can lead to $10-100\times$ improvements in MVM operations – *if the matrix fits within a single CIM tile or all parameters can be stored on-chip*. However the very factors that make CIM architectures efficient, namely parallel analog computing on the memory bitlines, also limit accuracy and performance such as reduced signal-to-noise ratios due to simultaneously accessing all memory cells in parallel, gradient optimization failure due to mismatch caused by memory element nonlinearity and other sources of error.

B. Digital Platforms

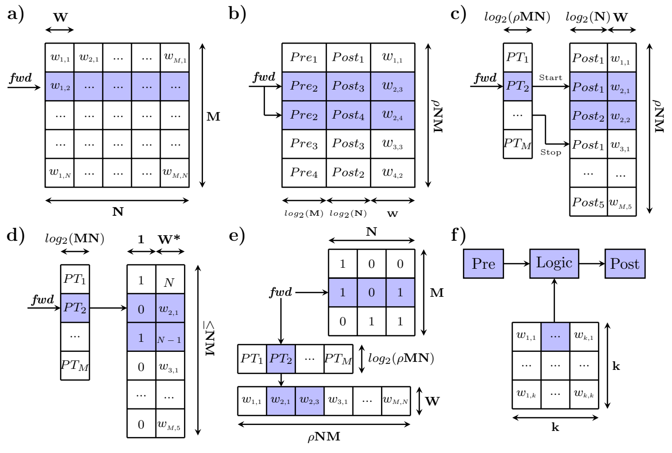

Fig. 2. a) crossbar memory organization (CB) [5], b) coordinate memory organization (COOR), c) pointer-based compressed sparse row memory organisation (PB-CSR) [4], d) pointer-based run-length encoded memory organisation (PB-RLE) - **W***: entries with run bit 0 require W bits meanwhile entries with run bit 1 need at max $log_2(N)$ bits [18], e) pointer-based bitmap memory organisation (PB-BMP) [19], f) functional memory organisation (FUNC) [20]. The highlighted section are part of a forward pass, while the transpose of the parameters must be accessed as during the backwards pass.

Most recent commercially available, large-scale SNN hardware have implemented fully digital designs [4], [5]. In such systems, the organization of synaptic weights and parameters can dramatically impact the system performance in terms of area, latency, and energy. Consider a system with multiple NSC, each containing M presynaptic neurons and N postsynaptic neurons. Fig. 2 provides an overview of various digital memory organization schemes that have been explored in literature [18]–[20]. In the figure, ρ models any sparsity in connectivity, primarily seen in dense/fully-connected layers. Convolution layers, are assumed to use square, unpruned, filters with the size given by k^2. Finally, each synapse is allocated W bits for parameters.

TABLE I
FULLY CONNECTED LAYER: MEMORY & ACCESS OVERHEAD.

		Forward	Backward
CB	Memory	MNW	MNW
	Overhead	RN	RN
PB-CSR	Memory	$M\log_2(\rho MN) + \rho MN(\log_2 N + W)$	$2(M\log_2(\rho MN) + \rho MN(\log_2 N + W))$
	Overhead	$R(1+\rho N)+CN$	$R(1+\rho N)+CN$
PB-RLE	Memory	$\leq M\log_2(MN) + NM(1+\hat{W})$	$\leq 2(M\log_2(MN) + NM(1+\hat{W}))$
	Overhead	$\leq R(1+N)$	$\leq R(1+N)$
PB-BMP	Memory	$M\log_2(\rho MN) + \rho MNW + MN$	$2(M\log_2(\rho MN) + \rho MNW) + MN$
	Overhead	$R(2+\rho N)$	$R(2+\rho N)$
In-Memory	Memory	NMW	NMW
	Overhead	RN	RN

TABLE II
CONVOLUTIONAL LAYER: # CONV. FILTERS OF SIZE Ck^2 & ACCESS OVERHEAD. GIVEN A CHIP WITH A BUDGET $\hat{N}MW$

		Forward	Backward
CB	# conv. filters	MN/k^2C	MN/k^2C
	Overhead	$R(\hat{N}+1)$	$R(\hat{N}+1)$
Func	# conv. filters	MN/k^2C	MN/k^2C
	Overhead	Rk^2+2Sk^2	Rk^2+2Sk^2
In-Memory	# conv. filters	MN/k^2C	MN/k^2C
	Overhead	$R(\hat{N}+1)$	$R(\hat{N}+1)$

IV. Design Considerations for SNN Hardware

Modern SNNs are composed of multiple specialized layers such as, pooling, convolutions, and fully connected (FC) layers. Since different SNN layer-types benefit from different memory organization [20], heterogeneity in available organization schemes improves energy efficiency. For example, FC layers in SNNs, are often overparameterized and can be pruned to 15% of the original model while maintaining accuracy [21]. Fig. 3 gives an overview of the energy-efficiency of three different memory organization schemes over a range of network sparsity (ρ), the crossbar [5], a pointer-based bitmap scheme [19], and the well known compressed sparse row (PB-CSR) scheme [18]. Depending on ρ for the layer and the activity in the network (low leakage contribution), different memory organization schemes are more energy-efficient. However,

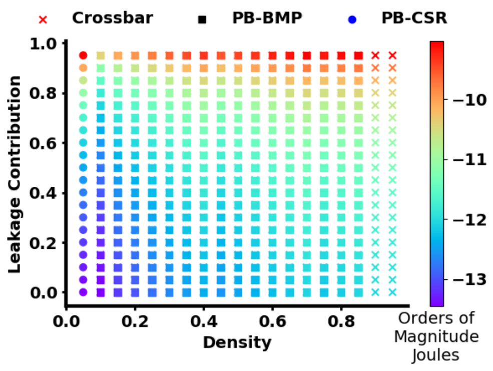

Fig. 3. Estimations of energy consumption for a fully connected layer (input 728, output 128) in an SNN with 8-bit weight precision. For the fraction of leakage energy contribution (y-axis) and the fraction of nonzeroes (ρ) in the layer (x-axis), we represent the estimated energy through the color and the most efficient storage scheme through the symbol. The energy is estimated for a combination of a single forward and backward pass as outlined in [20].

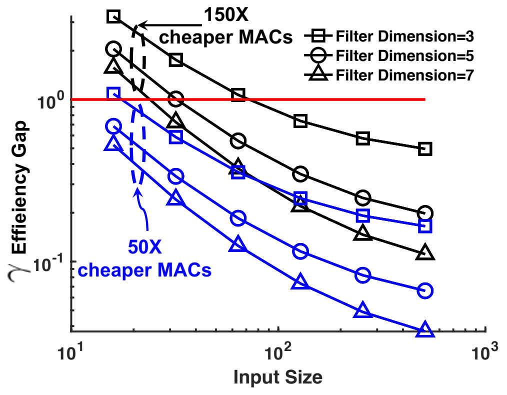

Fig. 4. Energy efficiency gap between CIM architectures and near-memory digital processors over a range of inputs. CIM architectures do not remain energy-efficient as the input and convolutional filter kernels are scaled. We model a constant 128×128 memory and include energy-costs of increased communication due to the “partial sum problem” for CIM cores.

note that neither CB, nor CIMs directly benefit from sparsity and their storage remains constant. Similar effects extend to convolutional layers. Depending on the layer size, CIM and CB memories might not be able to contain all the weights, requiring multiple NSCs to accumulate partial sums. Fig. 4 shows impact of scaling input and convolutional kernel size for a CIM architecture with size 128×128 when contrasted with near-memory digital designs. Although CIM architectures are extremely energy-efficient for small inputs, upon scaling, due to differential encoding of weights and inputs [16], they quickly lose efficiency compared to a traditional digital implementation that exploit structured computations.

Another important design consideration involves the trade-offs between the fidelity of implementing neural and synaptic temporal dynamics and the energy cost of implementing these temporal dynamics. As explained in Sec. II, these dynamics are central to the performance improvements of SNNs [21], [22]. Consequently, the hardware platform must support these time-dynamics to a sufficient degree. Fig. 5 illustrates the effect of quantizing neural dynamics (PQ) over a range of

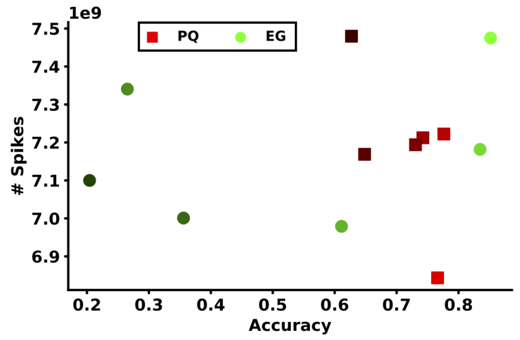

Fig. 5. Trade-off between quantization and spike-activity on the DVS gesture data set. Color shade represents the number of bits allocated to a variable, darker shades representing fewer bits and lighter shades more bits. The variables used in computing neuron dynamics are denoted by PQ and the variables used in gradient learning denoted by EG.

values (8 bits–14 bits), leading to an increase in spike activity as well as degrading SNN accuracy. Thus, any energy saved by avoiding temporal dynamics is immediately lost due to increased network activity. CIM structures incur a large energy cost when driving the memory array (see Fig. 1b). Driving the memory array with complex waveforms representing synaptic kernels, would dramatically limit the achievable energy-efficiency of a design. Thus, while CIMs might be amenable to simpler LIF neurons, currently, SRM models might be better suited to digital implementations. Of course, this is not to say that CIM architectures cannot implement temporal dynamics. As shown in [3], CIM-like implementations of SNNs are able to demonstrate a rich set of neural dynamics through analog delay. However, most existing CIM architectures are unsuited to implementing synaptic dynamics.

V. Outlook and Open Questions

We examined recent SNNs that use synaptic temporal dynamics for improved accuracy. The layer-types in these architectures are well suited to being implemented in digital systems. Additionally, although CIM based designs can be more computationally efficient, they are unsuited to implementing extended time-dynamics or exploiting the structured sparsity present in SNNs. However, the storage density and power savings offered by CIM architectures are critical to the design of next-generation energy-efficient, intelligent systems. Thus, for CIM architectures to truly deliver: (i) SNNs must be tailored to the CIMs fixed-dense block structure and (ii) alternative techniques enabling CIM structures to compute synaptic temporal dynamics must be employed. Jointly optimizing the memory subsystems together with the algorithm opens up a promising avenue to developing hardware that can match the computational efficiency of the biological brain.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[2] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor," *Frontiers in neuroscience*, vol. 7, p. 223, 2013.

[3] K. Bai, J. Li, K. Hamedani, and Y. Yi, "Enabling an new era of brain-inspired computing: Energy-efficient spiking neural network with ring topology," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[4] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[5] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[6] J. P. Mitchell, G. Bruer, M. E. Dean, J. S. Plank, G. S. Rose, and C. D. Schuman, "Neon: Neuromorphic control for autonomous robotic navigation," in *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*. IEEE, 2017, pp. 136–142.

[7] S.-C. Liu, B. Rueckauer, E. Ceolini, A. Huber, and T. Delbruck, "Event-driven sensing for efficient perception: Vision and audition algorithms," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 29–37, 2019.

[8] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, p. 73, 2011.

[9] S. Joshi, C. Kim, S. Ha, and G. Cauwenberghs, "From algorithms to devices: Enabling machine learning through ultra-low-power vlsi mixed-signal array processing," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2017, pp. 1–9.

[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[11] W. Gestner and W. Kistler, "Spiking neuron models," *Cambridge University*, 2002.

[12] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[13] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421.

[14] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," *arXiv preprint arXiv:1901.09948*, 2019.

[15] Y. Sakai, B. U. Pedroni, S. Joshi, A. Akinin, and G. Cauwenberghs, "Dropout and dropconnect for reliable neuromorphic inference under energy and bandwidth constraints in network connectivity," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2019, pp. 76–80.

[16] W. Wan, R. Kubendran, S. B. Eryilmaz, W. Zhang, Y. Liao, D. Wu, S. Deiss, B. Gao, P. Raina, S. Joshi, H. Wu, G. Cauwenberghs, and H. . P. Wong, "33.1 a 74 tmacs/w cmos-rram neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 498–500.

[17] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.

[18] B. U. Pedroni, S. Joshi, S. Deiss, S. Sheik, G. Detorakis, S. Paul, C. Augustine, E. O. Neftci, and G. Cauwenberghs, "Memory-efficient synaptic connectivity for spike-timing-dependent plasticity," *Frontiers in neuroscience*, vol. 13, p. 357, 2019.

[19] S. Joshi, B. U. Pedroni, and G. Cauwenberghs, "Neuromorphic event-driven multi-scale synaptic connectivity and plasticity," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 1–5.

[20] C. J. Schaefer, P. Faley, E. O. Neftci, and S. Joshi, "Memory organization for energy-efficient learning and inference in digital neuromorphic accelerators," *arXiv preprint arXiv:2003.11639*, 2020.

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

[22] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (decolle)," *arXiv preprint arXiv:1811.10766*, 2018.