Compiling Spiking Neural Networks to Mitigate Neuromorphic Hardware Constraints

Adarsha Balaji
Department of Electrical and Computer Engineering
Drexel University
Philadelphia, Pennsylvania, USA
adarsha.balaji@drexel.edu

Anup Das
Department of Electrical and Computer Engineering
Drexel University
Philadelphia, Pennsylvania, USA
anup.das@drexel.edu

***Abstract*—Spiking Neural Networks (SNNs) are efficient computation models to perform spatio-temporal pattern recognition on resource- and power-constrained platforms. SNNs executed on neuromorphic hardware can further reduce energy consumption of these platforms. With increasing model size and complexity, mapping SNN-based applications to tile-based neuromorphic hardware is becoming increasingly challenging. This is attributed to the limitations of neuro-synaptic cores, viz. a crossbar, to accommodate only a fixed number of pre-synaptic connections per post-synaptic neuron. For complex SNN-based models that have many neurons and pre-synaptic connections per neuron, (1) connections may need to be pruned after training to fit onto the crossbar resources, leading to a loss in model quality, e.g., accuracy, and (2) the neurons and synapses need to be partitioned and placed on the neuro-sypatic cores of the hardware, which could lead to increased latency and energy consumption. In this work, we propose (1) a novel unrolling technique that decomposes a neuron function with many pre-synaptic connections into a sequence of homogeneous neural units to significantly improve the crossbar utilization and retain all pre-synaptic connections, and (2) SpiNeMap, a novel methodology to map SNNs on neuromorphic hardware with an aim to minimize energy consumption and spike latency.**

***Index Terms*—Neuromorphic Computing, Spiking Neural Networks (SNNs), Machine Learning, Computation Graph.**

I. Introduction

SPIKING Neural Networks (SNNs) are machine learning approaches designed using spike-based communication and computation, and bio-inspired learning algorithms [1]. Neurons are typically implemented using Integrate-and-Fire [2] or Izhikevich [3] models. The neurons are interconnected using weighted synapses and communicate by sending short impulses, called spikes, over the synapse. Supervised, semi-supervised, or an unsupervised approach [4]–[6] can be employed to train the SNN. Information in SNNs can be represented as inter-spike interval or spike rates [7]–[9]. SNNs can be trained and inferred on the general purpose hardware [10], such as CPUs and GPUs. However, due to the *high* resource and energy demands, the use of this type of hardware in constrained environments is often *prohibitive*.

Neuromorphic hardware [11] (1) reduces the energy consumption significantly, due to their *low-power* design of neurons and synapses, and (2) improves application throughput, due to their *distributed* implementation of computation and storage. In recent years, several neuromorphic hardware platforms are designed: DYNAP-SE [12], TrueNorth [13], and Loihi [14]. Although these hardware differ in their operation (e.g., analog vs. digital), they all support neuro-synaptic cores realised using crossbar-based architectures. Multiple crossbars are interconnected over a time-multiplexed interconnect [15], [16].

A crossbar is a two-dimensional arrangement of synapses (n^2 synapses for n neurons). At the cross-point in the crossbar there is a synaptic element, which can be implemented using nano-scale Non-Volatile Memory (NVM) [17]. Within each crossbar, synaptic weights between pre- and post-synaptic neurons are programmed as conductance of NVM cells. Scaling up the size of a crossbar increases the number of synapses (n) per neuron, which exponentially increases the dynamic and leakage energy. To keep energy consumption reasonable, neuromorphic engineers limit the size of a crossbar. For instance, in TrueNorth and DYNAP-SE, the crossbar can accommodate a fixed number of synapses per neuron. To build large neuromorphic hardware, the common practice is therefore to integrate multiple crossbars, with the shared, time-multiplexed interconnect. While executing SNNs on crossbar-based neuromorphic hardware we encounter the following constraints:

Constraint 1: Crossbars on Neuro-synaptic cores can only accommodate a limited number of synaptic connections per post-synaptic neuron.

A $n \times n$ crossbar in a tile can accommodate only n pre-synaptic connections per post-synaptic neuron ($n = 128$ for the crossbars in DYNAP-SE). Model pruning is proposed as a solution. However, we observe that, even after model pruning, on average 2.6% of neurons in these models cannot be mapped to the crossbars in DYNAP-SE. There are two currently-used solutions to this problem – 1) implement larger crossbars, which increases the power consumption exponentially, and 2) remove synaptic connections, which reduces the model quality.

Constraint 2: The neurons and synapses of the SNN need to be efficiently partitioned and mapped to resources of the neuromorphic hardware.

Typically, the number of neurons and synapses of realistic SNN applications [18]–[20] far exceed the resources available on a single neuro-synaptic core (crossbar). Therefore, to execute this application on the neuromorphic hardware, the SNN first needs to be partitioned into clusters of neurons and synapses, where the intra-cluster *local synapses* are mapped within the crossbars, and the inter-cluster *global synapses* are mapped on the shared interconnect. An inefficient placement (ad-hoc) of the SNN on the crossbars of the hardware can increase energy consumption, and reduce application accuracy by increasing spike latency.

In this thesis, we address the constraints by:

- A novel unrolling technique [21] to decompose a neuron function with many pre-synaptic connections into a sequence of homogeneous neural units, where each neural unit is a function computation node, with two pre-synaptic connections.
- **SpiNeMap** [22]: an end-to-end methodology to partition an SNN into clusters of local and global synapses, and place the clusters on the crossbars, with an aim to minimize the energy consumption and increase the application accuracy.

II. Methodology and Results

A. Neuron Spatial Decomposition Methodology

We propose an unrolling approach [21], which decomposes a neuron function computation with many fan-ins into a sequence of homogeneous *neural units*, where each neural unit is a computation node with a maximum fanin-of-two (FIT). Here, one m-input neuron function is decomposed into $(m-1)$ two-input neural units connected in sequence.

The neuron function

$$y_o = \varphi\left(\sum_{i=1}^{m} n_i \cdot w_i\right) \tag{1}$$

is represented as

$$y_o = f(u_{m-1}),\ \text{where } u_i = \begin{cases} f(n_1 \cdot w_1 + n_2 \cdot w_2), & \text{for } i = 1 \\ f(u_{i-1} + n_{i+1} \cdot w_{i+1}), & \text{otherwise.} \end{cases} \tag{2}$$

where, f represents the neuron functionality of generating spike trains with a mean firing rate proportional to its input excitation, $n_1, n_2, \cdots, n_m$ are the m pre-synaptic neurons of the post-synaptic neuron n_o, and $w_1, w_2, \cdots, w_m$ are the corresponding synaptic weights. The total number of FIT neural units generated from a neural network with N neurons is

$$\mathcal{N} = \sum_{i=1}^{N}(m_i - 1),\ \text{where } m_i \text{ is the fan-in of neuron } n_i \tag{3}$$

B. Spatial Decomposition Results

The unrolling technique significantly improves crossbar utilization and ensures information integrity, resulting in no loss in model quality derived from connection pruning. We integrate this unrolling technique inside an existing SNN mapping framework and evaluate it using machine learning applications for a state-of-the-art neuromorphic hardware. Our results demonstrate an average 60% lower crossbar requirement, 9x higher synapse utilization, 62% lower wasted energy, and 3% increase in model quality compared to an existing SNN mapping approach.

C. SpiNeMap Methodology

The SpiNeMap methodology partitions SNNs into clusters of local synapses, and place these clusters on the physical crossbars such that energy consumption is minimized and application accuracy is increased. The contributions of SpiNeMap are as follows:

- **SpiNeCluster**: We propose a greedy heuristic to partition SNNs into local and global synapses with the objective of reducing the total number of global spikes.
  We represent the SNN as a set, G(N,S), of N neurons and S synapses. We are interested in partitioning this SNN into k clusters, such that the total number of inter-cluster spikes is minimized. The partitioning problem is treated as a graph partitioning problem [23], and has been applied in many context, including task mapping on multiprocessor systems [24]. The graph partitioning problem is already NP-complete [25], so heuristics are typically used to solve them [26]. We propose a greedy approach, roughly based on the Kernighan-Lin Graph Partitioning algorithm [23], which we show to be scalable to large SNNs.
- **SpiNePlacer**: We propose a PSO-based cluster placement approach to place the local synapses inside the physical crossbars, and time-multiplex the global synapses on the shared interconnect. The objective is to reduce energy consumption, spike latency, and spike disorder.
  PSO finds the optimum solution to a fitness function F. Each solution is represented as a particle in the swarm. Each particle has a velocity with which it moves in the search space to find the optimum solution. During the movement, a particle updates its position and velocity according to its own experience (closeness to the optimum) and also experience of its neighbors.
  The objective function F for SpiNePlacer, which is to minimize (1) energy consumption (E) on global synapses (i.e., the interconnect), (2) ISI distortion (I), and (3) spike disorder (D), for a given placement of the clusters in a partitioned SNN. We note that reducing the ISI distortion and spike disorder increases the application accuracy. We express SpiNePlacer's **objective function** as

$$F(\mathcal{P}_{\mathcal{H}}^{t}) = E \times I \times D, \tag{4}$$

  where $\mathcal{P}_{\mathcal{H}}^{t}$ is the placement of the partitioned SNN $\mathcal{H}(\mathcal{C}, \mathcal{E})$ at the t^{th} iteration of the PSO.

D. SpiNeMap Results

We show that SpiNeMap reduces energy consumption by 51% and spike latency by 17%, compared to the state-of-the-art techniques. Overall, SpiNeMap improves application accuracy between 1% and 26% (average 10%) for all the evaluated SNNs, executed on the DYNAP-SE platform.

III. Conclusion

In this thesis we propose a compiler framework for SNN-based application executed on tile-based neuromorphic hardware. The compiler performs a novel unrolling technique that decomposes a SNN neuron function with many pre-synaptic connections into a sequence of homogeneous neural units to significantly improve the crossbar utilization and retain all pre-synaptic connections followed by a novel methodology to map the neurons and synapses of the SNN on neuromorphic hardware with an aim to minimize energy consumption and spike latency. The proposed SNN compiler can be integrated with many other mapping approaches such as the data flow-based SNN compilation technique of [27]–[31], the circuit aging oriented SNN mapping technique of [32]–[36], and the SNN compiler of [37].

References

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, 1997.

[2] E. Chicca, D. Badoni, V. Dante *et al.*, "A vlsi recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory," *IEEE Transactions on Neural Networks*, vol. 14, 2003.

[3] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[4] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 6, pp. 902–918, 2001.

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

[6] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3227–3235, 2018.

[7] R. V. Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex," *Neural computation*, vol. 13, no. 6, pp. 1255–1283, 2001.

[8] A. D. Dorval, "Probability distributions of the logarithm of inter-spike intervals yield accurate entropy estimates from small datasets," *Journal of neuroscience methods*, vol. 173, no. 1, pp. 129–139, 2008.

[9] F. Peper, K. Leibnitz, J.-n. Teramae, T. Shimokawa, and N. Wakamiya, "Low-complexity nanosensor networking through spike-encoded signaling," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 49–58, 2016.

[10] A. Balaji *et al.*, "PyCARL: A PyNN interface for hardware-software co-simulation of spiking neural network," in *IJCNN*, 2020.

[11] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1367–1388, 2014.

[12] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[13] F. Akopyan, J. Sawada *et al.*, "TrueNorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[14] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[15] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *TBCAS*, 2017.

[16] A. Balaji, Y. Wu, A. Das, F. Catthoor, and S. Schaafsma, "Exploration of segmented bus as scalable global interconnect for neuromorphic computing," in *Great Lakes Symposium on VLSI*. ACM, 2019.

[17] A. Mallik, D. Garbin, A. Fantini, D. Rodopoulos, R. Degraeve, J. Stuijt, A. K. Das, S. Schaafsma, P. Debacker, G. Donadio, H. Hody, L. Goux, G. S. Kar, A. Furnemont, A. Mocuta, and P. Raghavan, "Design-technology co-optimization for OxRRAM-based synaptic processing unit," in *VLSIT*, 2017.

[18] A. Balaji, F. Corradi, A. Das, S. Pande, S. Schaafsma, and F. Catthoor, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *JOLPE*, 2018.

[19] A. Das, F. Catthoor, and S. Schaafsma, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ecg," *2018 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pp. 69–74, 2018.

[20] A. Das, P. Pradhapan, W. Groenendaal, P. Adiraju, R. T. Rajan, F. Catthoor, S. Schaafsma, J. L. Krichmar, N. Dutt, and C. Van Hoof, "Unsupervised heart-rate estimation in wearables with liquid states and a probabilistic readout," *Neural Networks*, 2018.

[21] A. Balaji, S. Song, A. Das, J. Krichmar, N. Dutt, J. Shackleford, N. Kandasamy, and F. Catthoor, "Enabling resource-aware mapping of spiking neural networks via spatial decomposition," *IEEE Embedded Systems Letters*, 2020.

[22] A. Balaji, A. Das, Y. Wu, K. Huynh, F. G. Dell'Anna, G. Indiveri, J. L. Krichmar, N. D. Dutt, S. Schaafsma, and F. Catthoor, "Mapping spiking neural networks to neuromorphic hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 76–86, 2019.

[23] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[24] A. Das, A. Kumar, and B. Veeravalli, "Communication and migration energy aware task mapping for reliable multiprocessor systems," *Future Generation Computer Systems*, vol. 30, pp. 216–228, 2014.

[25] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified np-complete problems," in *Proceedings of the sixth annual ACM symposium on Theory of computing*. ACM, 1974, pp. 47–63.

[26] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Design Automation Conference*. IEEE, 1982, pp. 175–181.

[27] S. Song, A. Balaji, A. Das, N. Kandasamy, and J. Shackleford, "Compiling spiking neural networks to neuromorphic hardware," in *The 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, 2020, pp. 38–50.

[28] A. Balaji, T. Marty, A. Das, and F. Catthoor, "Run-time mapping of spiking neural networks to neuromorphic hardware," *Journal of Signal Processing Systems*, pp. 1–10, 2020.

[29] A. Balaji and A. Das, "A framework for the analysis of throughput-constraints of snns on neuromorphic hardware," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2019, pp. 193–196.

[30] A. Das and A. Kumar, "Dataflow-based mapping of spiking neural networks on neuromorphic hardware," in *GLSVLSI*, 2018.

[31] A. Das, Y. Wu, K. Huynh, F. Dell'Anna, F. Catthoor, and S. Schaafsma, "Mapping of local and global synapses on spiking neuromorphic hardware," in *DATE*, 2018.

[32] T. Titirsha and A. Das, "Reliability-performance trade-offs in neuromorphic computing," *Computing with Unconventional Technologies (CUT)*, 2020.

[33] S. Song, A. Das, and N. Kandasamy, "Improving dependability of neuromorphic computing with non-volatile memory," *IEEE European Dependable Computing Conference (EDCC)*, 2020.

[34] T. Titirsha and A. Das, "Thermal-aware compilation of spiking neural networks to neuromorphic hardware," *Languages and Compilers for Parallel Computing (LCPC)*, 2020.

[35] A. Balaji, S. Song, A. Das, N. Dutt, J. Krichmar, N. Kandasamy, and F. Catthoor, "A framework to explore workload-specific performance and lifetime trade-offs in neuromorphic computing," *IEEE Computer Architecture Letters*, vol. 18, no. 2, pp. 149–152, 2019.

[36] S. Song and A. Das, "A case for lifetime reliability-aware neuromorphic computing," *International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2020.

[37] Y. Ji, Y. Zhang, W. Chen, and Y. Xie, "Bridge the gap between neural networks and neuromorphic hardware with a neural network compiler," in *ASPLOS*, 2018.