Performance and Energy Simulation of Spiking Neuromorphic Architectures for Fast Exploration

James A. Boyle[1], Mark Plagge[2], Suma G. Cardwell[2], Frances S. Chance[2], Andreas Gerstlauer[1]
[1]The University of Texas at Austin, Texas, USA
[2]Sandia National Laboratories, New Mexico, USA

ABSTRACT

Recent work in neuromorphic computing has proposed a range of new architectures for Spiking Neural Network (SNN)-based systems. However, neuromorphic design lacks a framework to facilitate exploration of different SNN-based architectures and aid with early design decisions. While there are various SNN simulators, none can be used to rapidly estimate latency and energy of different spiking architectures. We show that while current spiking designs differ in implementation, they have common features which can be represented as a generic architecture template. We describe an initial version of a framework that simulates a range of neuromorphic architectures at an abstract time-step granularity. We demonstrate our simulator by modeling Intel's Loihi platform, estimating time-varying energy and latency with less than 10% mean error for various sizes of a two-layer SNN.

KEYWORDS

Neuromorphic computing, performance modeling

ACM Reference Format:
James A. Boyle[1], Mark Plagge[2], Suma G. Cardwell[2], Frances S. Chance[2], Andreas Gerstlauer[1]. 2023. Performance and Energy Simulation of Spiking Neuromorphic Architectures for Fast Exploration. In *International Conference on Neuromorphic Systems (ICONS '23), August 1–3, 2023, Santa Fe, NM, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589737.3605970

1 INTRODUCTION

Neuromorphic computing takes elements inspired by the brain to accelerate and execute a range of applications, such as low-power real-time inference, solving optimization problems, and accelerating random-walk searches [1]. Spiking Neural Network (SNN) based designs are one major subclass of neuromorphic hardware. Spike-based computation is inherently event-driven, and has attractive properties such as being noise-tolerant and sparse in time and space. Spiking platforms accelerate the execution of SNNs, efficiently simulating the dynamics of spiking neurons and exchanging spike messages between weighted connections. Recently a range of different spiking platforms have been proposed and implemented. At the same time, the neuromorphic computing field is still evolving and an active research area, with new architectures (e.g., analog computing) being investigated. Despite this drive for developing new architectures, there is a lack of openly available tools for early design space exploration. During early design decisions, the designer must trade-off features based on required functionality while considering latency, energy and area costs for executing certain applications. Ideally, a designer should only have to consider architectural-level decisions without also having to consider low-level implementation details at this stage.

In this paper we show preliminary results for a high-level simulation framework that can rapidly estimate performance of different SNN-based platforms. Our framework is configurable to model a wide range of neuromorphic architectures, and we define a canonical file format for describing different designs. Our simulator implements a functional hardware model which uses a fast time-step based approach to simulate spiking designs. We combine simulated activity with per-update metrics to estimate the energy and latency for a user-specified SNN at each time-step. We demonstrate our simulator by modeling Intel's Loihi platform executing a real-world neuromorphic application. Results show that our simulator can accurately track trends and model Loihi's energy and latency with mean errors of 9.7% and 13.4% respectively.

2 RELATED WORK

When simulating SNNs, there are various levels of simulation fidelity that are applicable to different domains. Biologically-focused simulators, such as the NEural Simulation Tool (NEST) [7] and Brian2 [16] simulate the dynamics of abstract SNNs but do not model hardware implementations. When working at the hardware level, models of spiking platforms tend to abstract SNNs further. For example, simulators such as those in the Nengo [2] and Lava frameworks emulate spiking hardware at a functional level. They reproduce the functionality of one specific design while accounting for implementation effects such as variable bit-widths and quantization. However, these tools lack features to estimate latency and energy for different designs. Therefore, they are not helpful when exploring design trade-offs.

Hardware-focused simulation tools exist for spiking hardware, but these either focus on one aspect of the design or are otherwise limited. The simulator in [8] models networking on a spiking chip, but does not simulate performance of other hardware components. ATHENA [14] is an analytical tool that estimates energy for neuromorphic crossbar-based data-flow accelerators, but is not easily extendable to other spiking architectures. NeMo [13] and SST [15] use a discrete-event simulation model. SST models heterogeneous clusters of accelerators rather than a single architecture. NeMo was designed to simulate spiking architectures, but it does not estimate performance and focuses on a single platform. In contrast to such discrete-event models, our framework simulates designs at an abstract time-step granularity, without the need to model the precise timing of events for improved flexibility and simulator speed.

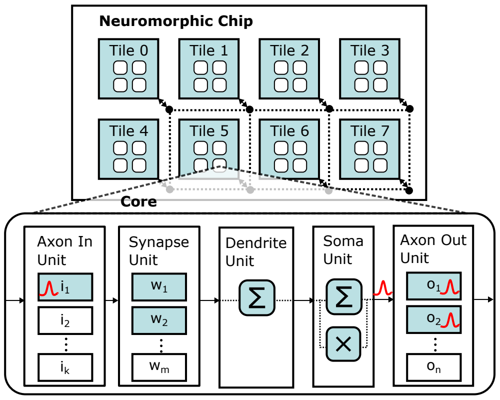
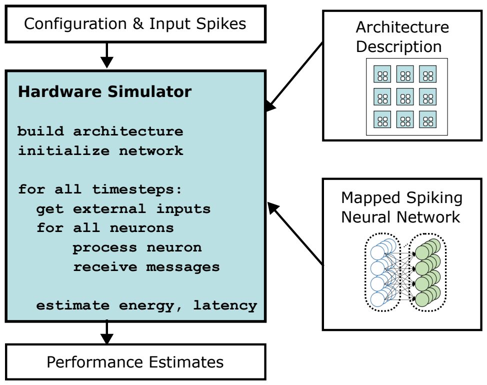


Figure 1: Large-scale spiking architecture template.



```
build architecture
initialize network

for all timesteps:
  get external inputs
  for all neurons
      process neuron
      receive messages

  estimate energy, latency
```

Figure 2: High-level overview of our framework. White boxes indicate custom file formats and inputs.

3 SPIKING HARDWARE ARCHITECTURES

Various spiking neuromorphic platforms, using either analog or digital designs, have been proposed (Table 1). Existing architectures generally follow a tile-based style with Network-on-Chip (NoC) or bus-based interconnect at varying scales in the number of tiles and cores per tile. Individual cores differ in their supported neuron models using either custom hardware or software implementations. Within each core, purely analog designs execute simulations in real-time using dedicated circuits to reproduce neuronal dynamics. By contrast, digital platforms operate in logical time and neurons access hardware in a time-multiplexed manner, sharing core resources. This work focuses on digital spiking platforms, and we do not consider purely analog designs.

Existing digital architectures have common design patterns, but differ in their implementation. Fig. 1 shows a generic architecture template for large-scale digital spiking platforms consisting of a grid of tiles each containing one or more cores connected via an NoC. Each core has a custom hardware pipeline to process a group of spiking neurons mapped to that core. Despite variations in implementation across designs, cores process spikes using a common sequence of neural-inspired operations. Spike messages are received over the network by a core at the input of an axon unit. The axon unit performs a lookup and generates a set of weight addresses. The synaptic unit then loads and processes each weight, filtering them according to a synaptic model before forwarding a current to the dendrite unit. The dendrite unit uses the connectivity and synaptic currents to compute and forward a single current to the soma unit. The soma unit in turn performs calculations to simulate the membrane potential, applying leaking and integrating the input current over time. If the membrane potential meets the threshold condition, a spike is sent to the axon output unit and the potential is reset. A spike at the output axon unit triggers a lookup of all destination cores to send spike messages to. Finally, the core sends one or more messages locally or globally, to be delivered to axons of connected neurons.

The digital platforms considered in this work operate in logical time, using a global time-step-based execution to ensure determinism and to enable neurons to share hardware resources. During each logical time-step, cores process state updates for their mapped neurons and synchronize using a global barrier. Spike messages that are generated during processing are broadcast and enter the pipelines at the receiving cores. Results are written to buffers in receiving cores, ready to be processed in the next time-step.

4 SIMULATOR DESIGN

An overview of our simulator is shown in Fig. 2. Our simulator uses the architecture template previously described to model performance and estimate latency and energy for a given SNN application executing on a modeled architecture. The framework takes a description of the hardware platform, a mapped SNN and run-time parameters passed on the command line. Using a custom simulator, we model spiking processing activity at a time-step granularity and use these activity counts to estimate energy and time-step latency.

4.1 Input Formats

We have created custom file formats for the architecture and SNN application description. The architecture description is YAML-based, defining a chip hierarchically from tiles, cores and functional units.

Table 1: Comparison of SNN-based hardware architectures.

Name	Neural Core Type	Cores Per Tile	Tile Count	Neuron Model	Interconnect
Loihi [4]	Custom Digital	4	32	Leaky-Integrate-and-Fire (LIF)	NoC Mesh
Loihi 2	Custom Digital	4	32	Any software-based	NoC Mesh
TrueNorth [5]	Custom Digital	1	4096	Augmented Integrate-and-Fire	NoC Mesh
SpiNNaker [6]	CPU-based	18	1	Any software-based	NoC
SpiNNaker2 [9]	CPU-based	4	38	Any software-based	NoC
Tianjic [12]	Custom Digital	1	156	LIF / sigmoid, tanh	NoC Mesh
Neurogrid [3]	Custom Analog	1	16	Ion channel model	Digital tree-based
BrainScaleS-2 [11]	Custom Analog	1	1	Adaptive Exponential	Digital bus

We define a set of keywords and parameters to define each hardware block, nesting cores inside tiles and defining the hardware units within each core. The architecture file also specifies energy and latency metrics for each operation and functional unit. The format is extensible, using lists of attribute-value pairs. Our format also supports replication of tiles and cores, enabling concise representation of designs while supporting heterogeneous configurations.

The mapped SNN is described using another custom file format, where each line describes either a neuron group, neuron, or neuron to core mappings. Groups represent populations of neurons that share common model parameters. Each neuron group contains one or more neurons, each defined by an identifier, its model parameters, and a list of its outgoing weighted connections. Finally, each mapping assigns a neuron to a hardware core in the design.

4.2 Time-step Based Simulation

We have implemented a custom simulator that loads a design and SNN from files, simulates the design at a time-step granularity and outputs statistics including latency and energy estimates. Neurons are loaded from the SNN file and stored as a list of objects, each object containing a neuron's state variables, mapped core and outgoing connections to other neurons. The hardware description is represented using classes for tiles and cores that store performance counters, cost metrics and the types of hardware unit supported. Different instances of each hardware unit are implemented as functions. Currently, we have implemented a limited number of neural hardware units, including a current-based synapse and a leaky-integrate-and-fire (LIF) soma model. In the future, other instances of each hardware unit will be added as a library of functions, chosen by attributes in the architecture description.

During each time-step we update neurons according to their soma and synaptic models. We first iterate over neurons, loading their previous dendritic current from a time-step buffer and calculating their state variable updates using an LIF model. If the neuron fired, we iterate over its outgoing connections and add the associated weight to the receiving neuron's dendritic current. This dendritic current is buffered, ready to be the input for the next time-step. We update counters for the number of soma updates, firing neurons and synaptic weights added. For firing neurons, we also model how many cores to send messages to, and count the total number of X and Y hops from all messages. Once all neurons are updated, counters are used to estimate energy and latency.

4.3 Energy and Latency Estimation

Performance counters in each core are combined with energy and latency cost metrics from the architecture configuration to estimate the total dynamic energy and time-step latency. These account for the cost of individual events in the hardware, such as soma updates, synaptic reads and sending a packet over the network. Total energy is calculated for each time-step by multiplying activity counts by update costs. We then sum energy over all activity types and cores.

Latency is calculated by considering the two processing stages of each core: message receiving and neuron processing (Fig. 3). Processing occurs in two stages that execute in parallel, separated by the time-step buffer: processing neurons mapped to that core and receiving spike messages from other cores. The buffer stores the results of the previous time-step i.e., inputs to the neuron processing

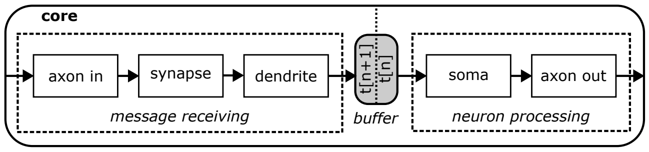


Figure 3: Spike processing pipeline model.

stage at the beginning of the time-step. The hardware units in each stage are determined by the position of the time-step buffer. In this initial implementation, we assume the time-step buffer to be before the soma, in future work we, will allow the buffer to be set at any position by the architecture description. Message receiving and neuron processing stages are processed in parallel within cores on the spiking chip. Our model calculates the latency for each stage on all cores, and the core latency as the maximum of its two stages. Network latency is accounted for in the neuron processing stage. The time-step latency is the maximum latency of any core.

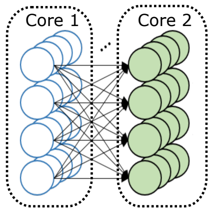
5 SIMULATOR RESULTS

For our experiments, we modeled Intel's Loihi platform [4] in our framework and adapted a two-layer SNN from a previously implemented power benchmark application [10]. Energy and latency metrics for Loihi were taken from published results in [4]. Fig. 4 shows activity counts for neuron updates and synapse reads per time-step, and the corresponding energy and latency (Fig. 4b) simulated for the benchmark SNN (Fig. 4a) with 1682 neurons and randomized spiking in the first layer. This demonstrates the capability of our time-step-based simulation approach to model time-varying activity and energy behavior, with counts and estimates calculated after every iteration. We show the two most significant activity counts for this application – neuron updates and synapse reads.

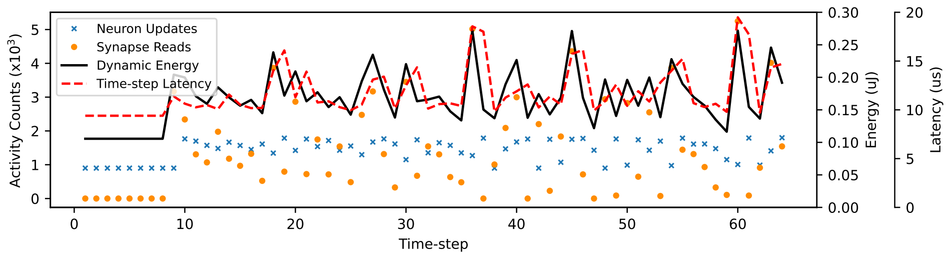
Fig. 5 shows energy and latency estimates over the aggregated execution of the benchmark SNN from Fig. 4(a) scaled to different numbers of neurons per layer and with all neurons in the first layer firing in every time step. Simulated estimates are compared against measurements from probes on an Intel Loihi-based Nahuku platform, executing the application SNN over 10^5 time-steps. We were able to estimate energy and time-step latency with a mean error of 9.7% and 2.5% respectively, and our simulator processed up to 11.7×10^6 spikes per second when run on an Intel i7-920 CPU. Fig. 6 shows latency estimates for the same network, partitioned across two cores to create more cross-core interactions. The events processed by the chip are the same for both mappings. As such, energy results matched Fig. 5 and are not shown here. For SNNs with fewer than 1024 neurons, the networks are mapped to a single Loihi core, processing all neurons serially. For networks larger than 1024 neurons, the second layer spills onto a second core, leading to parallelization of spike processing. As a result, fewer spikes are processed by the first core and the time-step latency is initially decreased. We were unable to accurately track latency in this region since our model does not yet account for cross-core interactions. Our network model assumes that all operations are parallelized whereas in reality, hardware contention prevents some spikes from being sent and processed immediately.

6 CONCLUSION AND FUTURE WORK

In summary, neuromorphic design-space exploration has a need for high-level, fast simulation of different spiking hardware platforms.

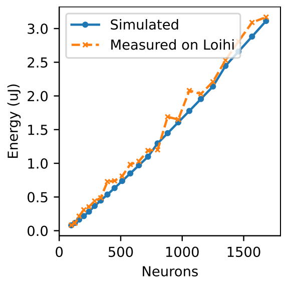


(a) Benchmark SNN

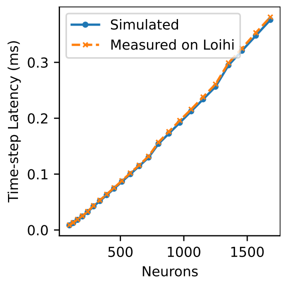


(b) Simulated activity, latency and energy estimates

Figure 4: Time-series showing total energy and latency estimates over each time-step for the benchmark SNN described in [10].

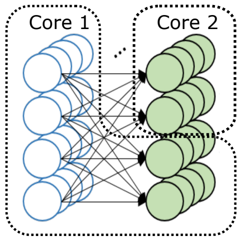


(a) Energy

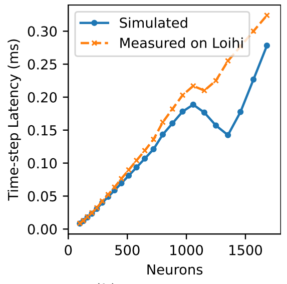


(b) Latency

Figure 5: Comparison of estimates for Loihi executing the two-layer benchmark SNNs from Fig. 4(a).



(a) Multi-core mapping



(b) Latency

Figure 6: Latency estimates for Loihi executing benchmark SNNs using the mapping shown in a). Energy results matched Fig. 5(a).

In this paper, we presented a framework to rapidly and accurately simulate latency and energy of spiking neuromorphic platforms. Our framework supports a range of architectures using a hierarchical file format following a generic architecture template. Our simulator models on-chip activity at an abstract time-step granularity. We use per time-step counts from simulation with cost metrics to estimate total energy and time-step latency. Initial results show that we can estimate time-varying energy and latency for Intel's Loihi with up to 90% accuracy for a realistic benchmark SNN.

Future work will develop a timing model to account for interactions and contention between cores to improve the latency estimation accuracy. We will also implement a library of hardware units to support other existing and future architectures. This will require adding new functions to support different neuron, synaptic and networking models. We plan to support both digital and analog elements, including components using beyond-CMOS devices. These will be modeled by simulating their dynamics for each time-step and finding an abstraction that captures their energy and latency while remaining fast for simulation. Furthermore, we will characterize our simulator's performance when modeling these devices and executing larger SNNs.

ACKNOWLEDGMENTS

This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan. SAND2023-06320C

REFERENCES

[1] J. Aimone et al. 2022. A review of non-cognitive applications for neuromorphic computing. *NCE* (2022).
[2] T. Bekolay et al. 2014. Nengo: a Python tool for building large-scale functional brain models. *Front. Neuroinform.* 7 (2014), 48.
[3] B. V. Benjamin et al. 2014. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 5 (2014), 699–716.
[4] M. Davies et al. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (2018), 82–99.
[5] M. V. DeBole et al. 2019. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer* 52, 5 (2019), 20–29.
[6] S. B. Furber et al. 2014. The SpiNNaker Project. *Proc. IEEE* 102, 5 (2014), 652–665.
[7] M. Gewaltig and M. Diesmann. 2007. NEST (NEural Simulation Tool). *Scholarpedia* 2, 4 (2007), 1430.
[8] R. et al. Kleijnen. 2022. A network simulator for the estimation of bandwidth load and latency created by heterogeneous spiking neural networks on neuromorphic computing communication networks. *JOLPE* 12, 2 (2022), 23.
[9] C. Mayr et al. 2019. Spinnaker 2: A 10 million core processor system for brain simulation and machine learning. *arXiv preprint arXiv:1911.02385* (2019).
[10] L. Parker et al. 2022. Benchmarking a Bio-inspired SNN on a Neuromorphic System. In *NICE*. 63–66.
[11] C. Pehle et al. 2022. The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity. *Front. Neurosci.* 16 (2022).
[12] J. Pei et al. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature* 572, 7767 (2019), 106–111.
[13] M. Plagge et al. 2018. NeMo: A Massively Parallel Discrete-Event Simulation Model for Neuromorphic Architectures. *TOMACS* 28, 4, Article 30 (Sept. 2018).
[14] M. Plagge et al. 2022. ATHENA: Enabling Codesign for Next-Generation AI/ML Architectures. In *ICRC*. 13–23.
[15] A. F. Rodrigues et al. 2011. The structural simulation toolkit. *ACM SIGMETRICS Performance Evaluation Review* 38, 4 (2011), 37–42.
[16] M. Stimberg et al. 2019. Brian 2, an intuitive and efficient neural simulator. *eLife* 8 (Aug. 2019), e47314.