Spiking Neural Networks for Near-Sensor Processing: An Open-Hardware Experience

Luca Martis[1], Gianluca Leone[1], Luigi Raffo[1] and Paolo Meloni[1]

[1]Department of Electrical and Electronic Engineering, University of Cagliari, 09123, Cagliari, Italy

Abstract

Moving data processing closer to its source, known as edge computing, offers significant advantages including reduced latency, bandwidth savings, and enhanced reliability. However, implementing complex algorithms on edge devices is challenging due to power and computational limitations. In this context, Spiking Neural Networks (SNNs) emerge as a promising solution due to their energy efficiency. This work presents a preliminary version of an accelerator for Spiking Neural Networks designed to be integrated into a System-on-Chip (SoC). The accelerator was developed using the open-source Process Design Kit (PDK) SKY130A and the open-source Electronic Design Automation (EDA) tool OpenLane. Preliminary results highlight the potential of SNNs for edge applications, as well as the advantages of using these new open-source tools, which reduce cost barriers and facilitate transparent information exchange.

Keywords

Edge computing, low power, spiking neural network, real-time, ASIC, EDA, open-source, OpenLane flow.

1. Introduction

In recent years, the deployment of artificial intelligence (AI) algorithms at the edge has gained significant traction, driven by the exponential growth of Cyber-Physical Systems (CPS) and Internet of Things (IoT) devices. Moving data processing as close as possible to the source offers significant benefits, such as the ability to perform real-time processing, reduce latency, and improve overall system efficiency. However, implementing these algorithms on edge devices is challenging due to constraints related to energy consumption and computational power.

Spiking Neural Networks (SNNs) have emerged as a promising AI algorithm for edge applications due to their energy efficiency. Unlike traditional neural networks, SNNs mimic the brain's spiking behavior, leading to lower power consumption and improved performance in event-driven scenarios. This makes SNNs particularly suitable for edge computing, where energy resources are constrained, and efficient processing is crucial.

These networks are ideally better performed on neuromorphic processors, which can fully exploit the sparsity of events and offer remarkable efficiency. Neuromorphic hardware is designed to support the unique operational characteristics of SNNs, such as asynchronous processing and event-driven computation, resulting in significant energy savings and faster response times. However, the adoption of neuromorphic processors is still limited due to high costs and integration challenges with existing edge devices.

In light of these considerations, the objective of this work is to develop a low-power accelerator for SNNs to be integrated into a System on Chip (SoC) for edge applications.

Leveraging current trends, we have utilized an open-source Process Design Kit (PDK) and open-source Electronic Design Automation (EDA) tools to implement the system. This approach not only aligns with the growing emphasis on open-source solutions but also enhances the flexibility, accessibility, and cost-effectiveness of our design process.

The main findings of this study can be summarized as follows:

CPS summer school

luca.martis@unica.it (L. Martis); gianluca.leone94@unica.it (G. Leone); raffo@unica.it (L. Raffo); paolo.meloni@unica.it (P. Meloni)

0009-0002-6501-9506 (L. Martis); 0000000152650759 (G. Leone); 000000019683009X (L. Raffo); 0000000281064641 (P. Meloni)

- we present an SNN accelerator integrable into a SoC, such as the Efabless Caravel[1];
- we present an initial version of the accelerator layout, created using exclusively open-source tools;
- we have demonstrated that the SNN accelerator is a viable solution for edge applications, achieving an energy consumption of 3.7 μJ per inference at a frequency of 89 MHz.

The remainder of the paper is organized as follows: Section 2 provides a brief overview of related works, Section 3 provides an introduction to spiking neural networks, the architecture of the accelerator, and the RTL-GDSII flow. Finally, Section 4 discusses the obtained results, while Section 5 outlines future work, and Section 6 presents the conclusions.

2. Related Works

Currently, several specialized neuromorphic processors are highly effective in executing SNNs. These advanced processors, as detailed in [1], [2], and [3], are capable of efficiently handling large-scale networks while maintaining high energy efficiency. However, these devices are not suitable for implementation on edge devices, as they require power levels ranging from hundreds of milliwatts to Watts, which are not manageable by such devices.
More relevant to this work are the studies [4] and [5], which focus on implementing FPGA-based accelerators for lightweight SNNs, making them better suited for edge applications. Particularly relevant to this study is the work described in [4], where a previous version of the accelerator used in this study was implemented on an Artix-7 FPGA.
To the best of our knowledge, there are no works that describe accelerators for SNNs developed entirely using open-source tools. However, several studies have explored the potential of these emerging tools and demonstrated their capabilities in various applications.
The studies by [6] and [7] are particularly relevant to our work, as they use OpenLANE [8] and the Sky130A PDK to implement an accelerator integrated into an SoC. The research in [6] involved designing a Vector Accelerator Unit for integration with the Caravel SoC, aimed at enhancing the performance of the existing RISC-V processor on the board. The results showed significant performance improvements, with parallel processing speeds ranging from 0.19 to over 100 times faster for (12 × 12) Scalar Products.
In [7], an RRAM-based In-Memory Computing (IMC) co-processor is introduced, which enables energy-efficient mapping of Multiply and Accumulate (MAC) operations and offers reconfigurable logic mapping.
The work presented in [9] describes the design of a 3-stage pipelined RISC-V RV32I processor using the OpenLANE tool in conjunction with the open-source Sky130A PDK. This study compared the results obtained using open-source EDA tools with those from commercial tools, demonstrating that the timing, area, and power consumption metrics from the OpenLANE flow were competitive and efficient relative to commercial flows.
Finally, the study in [10] represents the most complex project, utilizing OpenRoad [11] and Yosys [12] tools along with the open-source PDK IHP-SG13 to develop Basilisk, the first fully open-source, Linux-capable RISC-V SoC. Basilisk features a 64-bit RISC-V core, a fully digital HyperRAM DRAM controller, and a comprehensive set of I/O peripherals, including USB 1.1 and VGA. This work highlights the advantage offered by open-source tools to better customize the provided scripts. by modifying the Yosys and OpenRoad scripts, we achieved better results compared to using the non-customized scripts, including higher system clock frequencies and reduced area occupancy.

3. Methods

3.1. Spiking Neural Network

Spiking neural networks are defined as the third generation of neural networks. What distinguishes them from other conventional neural networks is their use of a more realistic neuron model. Neurons

[1] https://caravel-harness.readthedocs.io/en/latest/

communicate with each other through binary signals known as spikes and maintain an internal state called membrane potential.
The neuron model used in this work is the Leaky Integrate and Fire (LIF) neuron, which is described by the following equation:

$$\begin{aligned} U[t] &= \beta U[t-1] + \sum ws[t] - S_{out}[t-1]\theta \\ S_{out}[t] &= \begin{cases} 1, & \text{if } U[t] > \theta \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

where U represents the neuron's membrane potential, β is the membrane potential decay rate, w denotes the synaptic weights, s indicates the input spikes, S_{out} is the output spike, and θ represents the neuron's threshold. When the potential exceeds the threshold value θ, a spike S_{out} is generated and a reset mechanism is activated, subtracting the threshold value from the potential.
The communication through spikes makes these networks particularly interesting from an energy standpoint for several reasons. Spiking neural networks operate in an event-driven manner, where signals remain inactive for extended periods and only become active when spikes occur. This phenomenon, known as sparsity, allows the system to process information only when necessary, thus avoiding unnecessary operations and conserving energy during inactive periods. Additionally, since the inputs to these neurons are binary (0 or 1), the computational operations primarily involve simple accumulation rather than complex multiplications. This reduces the overall algorithmic complexity and leads to lower energy consumption compared to conventional neural networks, which rely heavily on multiplication operations. These characteristics make spiking neural networks highly efficient and well-suited for edge applications where energy efficiency is critical.

3.2. System Architecture

The proposed system is an accelerator for Spiking Neural Networks intended for integration into a System-on-Chip (SoC). This version of the accelerator is a modified adaptation of the one described in [4], which was initially designed for FPGA implementation.
As the original design leveraged FPGA-specific components like configurable blocks of memory (BRAMs) and hard-wired Digital Signal Processor (DSP), several modifications were required to adapt the architecture to the PDK used. Modules previously implemented with DSPs were replaced by combinational logic, while blocks originally implemented with BRAMs were substituted with fourteen 4 Kbyte single-port SRAM memories [2] and flip-flops.
In particular, the weight memories were implemented using eight memories, while the spike memories were realized using flip-flops, as this approach proved more area-efficient due to the mismatch with the available memory blocks. In the FPGA, the BRAMs were dual-port, allowing simultaneous read and write operations. Therefore, the FIFOs for membrane potentials were implemented to support concurrent read and write access. However, since the available memories are now single-port, it was necessary to adapt the architecture by using two separate memories and a state machine. During each inference, one memory is used for reading and the other for writing, swapping roles in the subsequent inference. Finally, two memories were used to store the decay factors. Weights are stored in 8-bit integer format, membrane potentials in 32-bit format to avoid saturation, and decay factors in 14-bit format.
The accelerator receives input spikes in groups of four, while the membrane voltages of the neurons in the final layer of the network are used as output.
The accelerator executes a network with four dense layers, using two layer hardware modules on which two layers are mapped each. Each module contains a memory that stores the synaptic weights, a FIFO where the neurons' decay factors are saved, and a FIFO where the neurons' membrane potentials are saved. Each layer also includes two stacks that dynamically address the active input synapses and two memories that store the generated spikes. Additionally, every layer contains a module that calculates

[2] https://github.com/efabless/EF_SRAM_1024x32

the synaptic current and a neuron module for updating the LIF neuron state.
The hardware layer module operates as follows: the module receives input spikes, grouped in sets of four, and stores them in the spike memory 0, keeping track of the addresses of the active groups in the spike stack 0. The addresses saved in stack 0 are used to access the weight memory, where the weights are also stored in groups of four and are summed based on the spike condition of the associated synapses stored in spike memory 0. The LIF modules update the potential of the neurons and generate the spikes for the next layer. The generated spikes are again grouped in sets of four and stored in spike memory 1, and the active addresses are saved in stack 1. This process is then repeated, and the generated outputs become the input for the next layer module.

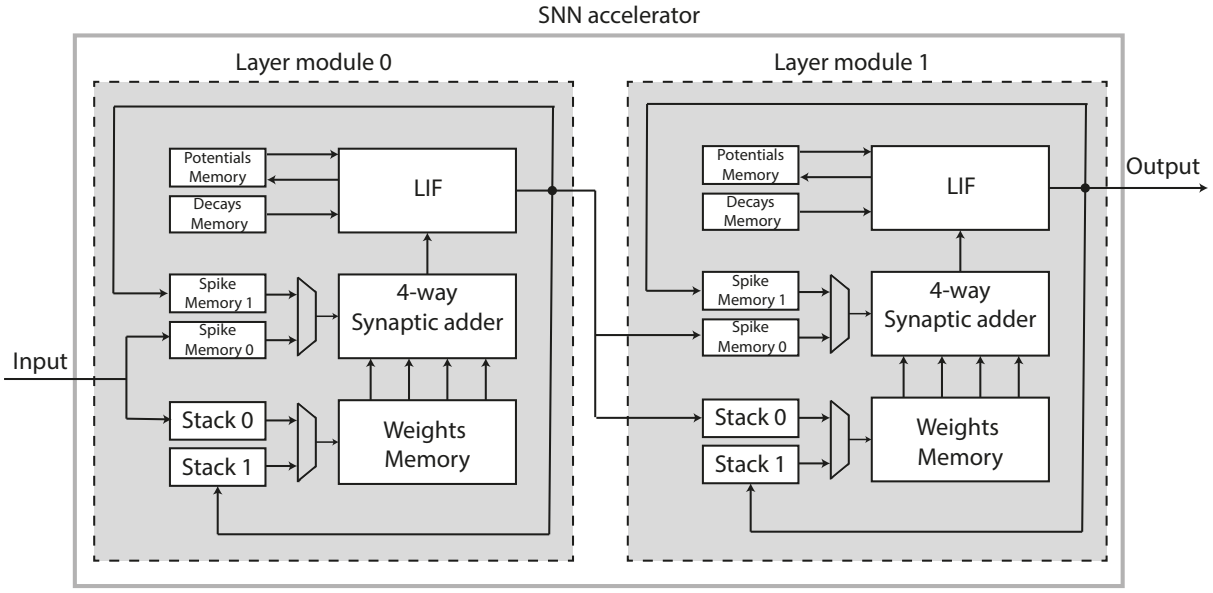


Figure 1: Proposed System Architecture. Each layer module manages two layers of the network. It features memory for storing the weights of these layers, a synaptic adder for accumulating synaptic weights, and a neuron module for updating the LIF neuron states. Additionally, each subprocessor includes two spike memories for buffering partial results from layer executions and two spike stacks for tracking active spikes.

3.3. RTL-GDSII flow

To develop the chip layout of the proposed system, the open-source OpenLANE flow [8] was used.
This tool is an automated RTL to GDSII flow designed for implementing single macros, which will then be integrated into SoCs, and complete chips. In particular, the tool is maintained by EFabless, and its main advantage is the ability to integrate the designs produced into a system that includes essential functions such as IO, power management, and configuration called Caravel.
The flow relies entirely on open-source tools: Yosys [12] for synthesis, OpenSTA [3] for static timing analysis, and OpenROAD [11] for floorplanning, placement, CTS, and routing. KLayout [4] generates the final GDSII layout file, while Magic [5] and Netgen [6] are used for Design Rule Checks (DRC) and Layout Versus Schematic (LVS) verification.
The tool is compatible with both A and B variants of the open-source SKY130 PDK, the C variant of the open-source GF180MCU PDK, and includes documentation for adding support for other PDKs, including proprietary ones. For this work, version A of the open-source SKY130 PDK was selected, to enable the integration of the system onto Caravel.

[3] https://github.com/The-OpenROAD-Project/OpenSTA
[4] https://www.klayout.de/
[5] https://github.com/RTimothyEdwards/magic
[6] https://github.com/RTimothyEdwards/netgen

The flow is completely automated and requires only the RTL files describing the system and a configuration file, in either JSON or TCL format, which outlines the settings to follow.

4. Results

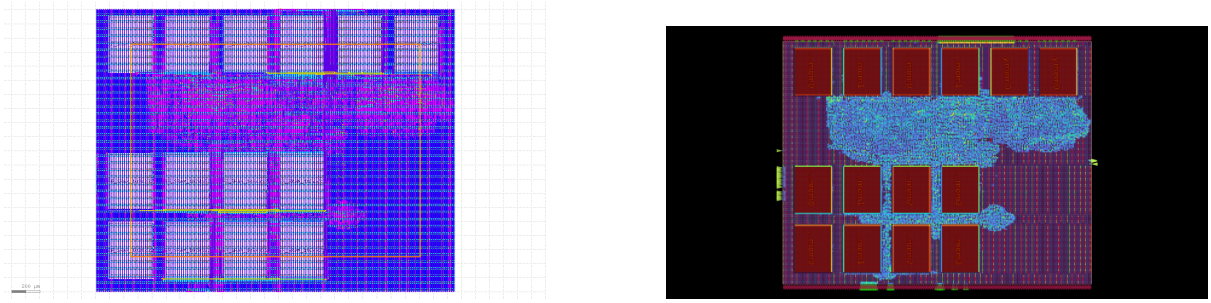
The table 1 presents the main statistics of the chip. In the figure 2, the layout visualized with Klayout is shown on the left, while the placement density visualized through the OpenROAD graphical interface is shown on the right. The core area of the chip was chosen to meet the constraints imposed by the Caravel Harness and occupies 46% of the available area, leaving 5.7 mm² of space for additional modules. The system can operate at a maximum frequency of 89 MHz, dissipating a power of 54.3 mW at this frequency, as calculated using the OpenSTA tool.
Since the hardware can perform 4-synaptic additions per clock cycle, one inference can be executed every T_{inf}, expressed as

$$T_{inf} = \frac{N_{\text{synapse}}}{4} \cdot T_{clk} \tag{2}$$

where N_{synapse} is the number of active synapses and T_{clk} is the clock period.
In the worst-case scenario, where all the synapses are active, T_{inf} is equal to 70 μs and energy dissipated per inference is equal to 3.7 μJ.

Table 1
Key metrics of the SNN accelerator

Core Area	2550x2000 μm
Combinational cells	15073
Sequential cells	2092
SRAM memory	57KB (14 macros)
F_{max}	89 MHz
Total Power	54.3 mW



(a) Final layout (b) Placement density

Figure 2: Design layout using OpenLane flow. At the bottom left, we can see the eight memories related to the weights. At the top, we can see the six memories related to the storage of decay factors and membrane potential.

The results obtained are comparable to those in [4], where a similar version of the accelerator presented in this work was used, achieving a power dissipation of 56.4 mW at a frequency of 80 MHz. However, it should be noted that this power dissipation includes the entire system, not just the accelerator, which was equipped with a Microblaze softcore and a signal encoding module.

5. Future works

A fundamental aspect of using spiking neural networks is the process of converting inputs from real values to spike trains. Conversely, the process of converting the output spikes from the network back into real values is equally important. In the proposed accelerator, it is assumed that the encoding phase

has already been completed, with spikes being received directly as input, while the decoding phase is not necessary as the output is the membrane potential, which is already a real value.
In the future, the goal is to add spike encoding and decoding modules to make the system as complete as possible. Various conversion algorithms have been proposed in the literature for the conversion task, including Ben's Spiker [13], step-forward [14] or Pulse width modulated-Based [15], which are better suited depending on the use case. To accommodate different applications, the plan is to integrate an embedded FPGA (eFPGA) into the accelerator. This will allow for the mapping of various encoding and decoding algorithms, depending on the application, leveraging the flexibility of FPGA reconfigurability while maintaining the low power consumption of an ASIC. This integration will be facilitated by new open-source tools, including OpenFPGA [16] and FABulous [17], which enable the generation of RTL to describe the FPGA and provide the necessary tools to generate the bitstream for configuring them. Once the eFPGA is integrated into the system, the final goal will be to perform the tapeout of the chip.

6. Conclusion

In this work, we presented an accelerator for SNNs designed for integration into a SoC. An initial version of the accelerator layout was provided, created entirely using open-source tools and compatible with integration into the Caravel SoC. Additionally, the energy efficiency of the accelerator was demonstrated, with an energy consumption of 3.7 μJ per inference at a frequency of 89 MHz.
As a preliminary step, these results highlight the potential for further improvements in both energy efficiency and area. The customization possibilities offered by open-source tools present a promising path for future optimization, suggesting that our approach could evolve into more competitive and scalable solutions in the field of hardware accelerators for SNNs.

References

[1] G. Tang, A. Shah, K. Michmizos, Spiking neural network on neuromorphic hardware for energy-efficient unidimensional slam, 2019, pp. 4176–4181. doi:10.1109/IROS40897.2019.8967864.

[2] M. V. DeBole, B. Taba, A. Amir, F. Akopyan, A. Andreopoulos, W. P. Risk, J. Kusnitz, C. O. Otero, T. K. Nayak, R. Appuswamy, et al., Truenorth: Accelerating from zero to 64 million neurons in 10 years, Computer 52 (2019) 20–29.

[3] J. Behrenbeck, Z. Tayeb, C. Bhiri, C. Richter, O. Rhodes, N. Kasabov, J. I. Espinosa-Ramos, S. Furber, G. Cheng, J. Conradt, Classification and regression of spatio-temporal signals using neucube and its realization on spinnaker neuromorphic hardware, Journal of Neural Engineering 16 (2019) 026014. URL: https://dx.doi.org/10.1088/1741-2552/aafabc. doi:10.1088/1741-2552/aafabc.

[4] G. Leone, L. Martis, L. Raffo, P. Meloni, Spiking neural networks for integrated reach-to-grasp decoding on fpgas, in: 2023 IEEE Biomedical Circuits and Systems Conference (BioCAS), 2023, pp. 1–5. doi:10.1109/BioCAS58349.2023.10389037.

[5] A. Carpegna, A. Savino, S. Di Carlo, Spiker: an fpga-optimized hardware accelerator for spiking neural networks, in: 2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2022, pp. 14–19. doi:10.1109/ISVLSI54635.2022.00016.

[6] E. I. Baungarten-Leon, S. Ortega-Cisneros, U. Jaramillo-Toral, F. J. Rodriguez-Navarrete, L. Pizano-Escalante, J. J. R. Panduro, Vector accelerator unit for caravel, IEEE Embedded Systems Letters 16 (2024) 73–76. doi:10.1109/LES.2023.3267341.

[7] V. Parmar, A. Ray, C. Moorthii, R. Mishra, D. Verma, D. Pandey, M. Suri, Open-rimc: Open-source rram-based imc co-processor with reconfigurable logic mapping, in: 2023 IEEE 23rd International Conference on Nanotechnology (NANO), 2023, pp. 519–523. doi:10.1109/NANO58406.2023.10231247.

[8] M. Shalan, T. Edwards, Building openlane: A 130nm openroad-based tapeout- proven flow : Invited paper, in: 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2020, pp. 1–6.

[9] S. Hesham, M. Shalan, M. W. El-Kharashi, M. Dessouky, Digital asic implementation of risc-v: Openlane and commercial approaches in comparison, in: 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2021, pp. 498–502. doi:10.1109/MWSCAS47672.2021.9531753.

[10] P. Scheffler, P. Sauter, T. Benz, F. K. Gürkaynak, L. Benini, Basilisk: An end-to-end open-source linux-capable risc-v soc in 130nm cmos, 2024. URL: https://arxiv.org/abs/2406.15107. arXiv:2406.15107.

[11] D. B. T. Ajayi, Openroad: Toward a self-driving, open-source digital layout implementation tool chain, Proceedings of Government Microcircuit Applications and Critical Technology Conference (2019). URL: https://par.nsf.gov/biblio/10171024.

[12] D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist, M. Milanovic, Yosys+ nextpnr: an open source framework from verilog to bitstream for commercial fpgas, in: 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), IEEE, 2019, pp. 1–4.

[13] B. Schrauwen, J. Van Campenhout, Bsa, a fast and accurate spike train encoding scheme, in: Proceedings of the International Joint Conference on Neural Networks, 2003., volume 4, 2003, pp. 2825–2830 vol.4. doi:10.1109/IJCNN.2003.1224019.

[14] K. Wang, X. Hao, J. Wang, B. Deng, Comparison and selection of spike encoding algorithms for snn on fpga, IEEE Transactions on Biomedical Circuits and Systems 17 (2023) 129–141. doi:10.1109/TBCAS.2023.3238165.

[15] A. Arriandiaga, E. Portillo, J. I. Espinosa-Ramos, N. K. Kasabov, Pulsewidth modulation-based algorithm for spike phase encoding and decoding of time-dependent analog data, IEEE Transactions on Neural Networks and Learning Systems 31 (2020) 3920–3931. doi:10.1109/TNNLS.2019.2947380.

[16] X. Tang, E. Giacomin, B. Chauviere, A. Alacchi, P.-E. Gaillardon, Openfpga: An open-source framework for agile prototyping customizable fpgas, IEEE Micro 40 (2020) 41–48. doi:10.1109/MM.2020.2995854.

[17] D. Koch, N. Dao, B. Healy, J. Yu, A. Attwood, Fabulous: An embedded fpga framework, in: The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 45–56. URL: https://doi.org/10.1145/3431920.3439302. doi:10.1145/3431920.3439302.