SYNtzulA: Open Hardware for Near-Sensor SNN Inference

Luca Martis, Gianluca Leone, Luigi Raffo, Paolo Meloni

***Abstract*—Spiking Neural Networks (SNNs) exploit event-driven processing to offer high energy efficiency when deploying Artificial Intelligence (AI) on wearable edge devices. However, specialized hardware is needed to fully take advantage of this potential, which, despite recent advances, remains expensive and not widely accessible. To address this, open-source Electronic Design Automation (EDA) tools and Process Design Kits (PDKs) offer a path to democratize the development of neuromorphic hardware. In this work, we present SYNtzulA, a system-on-chip designed for SNN acceleration, developed using the open-source IHP-SG13G2 130 nm PDK and the OpenROAD toolchain. The chip integrates a RISC-V softcore and a dedicated SNN accelerator, occupying approximately 6.8 mm^2 including I/O pads. It operates at up to 125 MHz, reaching a throughput of 2 Giga Synaptic Operations per second (GSOP/s) with an energy consumption of 36.5 pJ per synaptic operation. The accelerator can exploit the sparsity of spike-based computation by skipping unnecessary operations, resulting in total energy consumption in the order of a few hundred nanojoules per inference in different use cases involving biosignal analysis.**

***Index Terms*—Spiking Neural Networks, Open Hardware, ASIC, Low Power**

I. Introduction

Spiking Neural Networks are a computational model inspired by the structure and functioning of biological neural networks. As in traditional artificial neural networks (ANNs), SNNs are built from basic computational units, i.e. neurons, that are interconnected through synaptic weights to form complex network architectures. A key distinction of SNNs is that they process information through sparse binary signals, called spikes, in an event-driven manner. This operational model drastically reduces the number of required computations, making SNNs particularly well-suited for applications demanding high energy efficiency. Moreover, the spike-based computation in SNNs replaces the energy-intensive multiply-and-accumulate operations typical of ANNs with simpler addition operations, further enhancing their energy efficiency. Thanks to these characteristics, SNNs are particularly well-suited for wearable edge processing devices, where both energy and computational resources are severely constrained. However, to fully exploit the benefits of SNN algorithms, dedicated hardware is essential. The execution of SNNs achieves its highest efficiency on specialized neuromorphic processors designed to leverage temporal sparsity and minimize energy consumption per operation. In recent years, several of these processors have demonstrated exceptional performance in efficiently running large-scale SNN models [1]–[4]. Despite the significant promise of neuromorphic systems, their widespread deployment is currently constrained by prohibitive development costs and restricted accessibility.

Recent proposals, notably those by [5]–[7], have investigated Field-Programmable Gate Array (FPGA)-based architectures to overcome some of the challenges associated with neuromorphic hardware, focusing primarily on image classification tasks with the goal of achieving high throughput. FPGA solutions provide a straightforward and cost-effective way to implement neuromorphic systems, supporting the objective of democratizing access to this technology. Furthermore, the study in [8] has placed greater emphasis on deep edge and sensor processing applications, demonstrating the effectiveness of this approach within these domains [9]–[12].

However, a major limitation of FPGAs is their relatively high static power consumption compared to Application-Specific Integrated Circuits (ASICs) [13]. A significant portion of this static power is dissipated by the routing infrastructure itself [14], regardless of the actual logic utilization, making it difficult to benefit from the event-driven sparsity typical of SNN workloads by setting the system in idle mode.

An alternative that may ease access to neuromorphic computing for small-scale production while overcoming the limitations of FPGAs, is offered by the growing ecosystem of open-source EDA tools and open PDKs, which enable full digital ASIC design without relying on commercial toolchains or hardly accessible technologies. Although current open-source PDKs are primarily based on legacy technology nodes, falling short of the performance and integration levels offered by modern manufacturing processes, they still provide significant flexibility of use and enable fine tuning of power-related features. Thus, such libraries mark a significant step toward the creation and democratization of custom, efficient neuromorphic hardware, with an increasing impact as more advanced open-source PDKs become available.

In this work, we leverage a recent entirely open-source development flow, to design and implement a fully digital SNN accelerator optimized for lightweight networks, with the goal

This manuscript is submitted as an extended version of the paper titled “SYNtzulA: Open-Source Hardware for Energy-Efficient Spiking Neural Network Inference” by L. Martis, G. Leone, L. Raffo, and P. Meloni, originally presented at the 22nd ACM International Conference on Computing Frontiers (CF’25).

This work was funded by the projects EdgeAI “Edge AI Technologies for Optimised Performance Embedded Processing” and H2TRAIN ”Enabling digital technologies for Holistic Health-lifestyle motivational and assisTed supeRvision supported by Artificial Intelligence Networks”, supported by the Chips Joint Undertaking and its members including top-up funding by Austria, Belgium, France, Greece, Italy, Latvia, Netherlands, and Norway under grant agreement No 101097300 and No 101140052. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Chips Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them. It is also partially funded by NextGenerationEU Mission 4, Component 2, Investment 1.5, CUP B83C22002820006—Project METBIOTEL—Innovation Ecosystem ECS 0000024 ROME TECHNOPOLE SPOKE 1, and SPOKE 6.

The authors are with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123, Cagliari, Italy (email: luca.martis@.unica.it, gianluca.leone94@unica.it, raffo@unica.it, paolo.meloni@unica.it)

of optimizing sparsity exploitation for ultra-low-power near-sensor processing.

Building upon the open-source neuromorphic processor for sensor data presented in [8], in this work we introduced the following main contributions:

- We present an adaptation of the system described in [8], originally designed for implementation on a low-power FPGA, to the open-source IHP-SG13G2 PDK [1].
- We extended the dual-core architecture introduced in [8], leveraging its scalability and the customizability enabled by ASIC design to implement a quad-core version with higher throughput and improved energy efficiency.
- We validate our approach by comparing it with the low-power FPGA implementation, achieving higher energy efficiency and throughput while demonstrating that the use of open-source EDA tools and PDKs can represent a viable alternative to FPGAs for democratizing access to neuromorphic hardware.
- We compared our accelerator with another SNN implementation also developed using open-source EDA tools and PDKs, showing superior performance in terms of flexibility and resource utilization.

The aim of this work is to demonstrate that an extremely low-resource design can still deliver meaningful acceleration for sparse workloads, showing that sparsity-related power savings can be effectively exploited even in an older and open-source technology node and within an open-source design flow. In doing so, the proposed approach aims to achieve a balance between the accessibility typical of FPGA-based solutions and the efficiency offered by ASIC implementations, thus providing a path toward democratizing access to neuromorphic hardware. The remainder of the paper is organized as follows. Section II reviews the state of the art of neuromorphic hardware accelerators. Section III introduces the proposed system architecture, while Section IV describes the backend flow adopted for the ASIC implementation. Section V outlines the design process and the optimizations that transformed the original FPGA-based implementation into an ASIC version, emphasizing how the latter more efficiently exploits available hardware resources. Section VI presents the experimental results, focusing on key metrics such as area, timing, and power consumption. Section VII complements this analysis by comparing our ASIC design with both the FPGA implementation and an existing open-source SNN accelerator. Finally, Section VIII discusses the limitations of this work and outlines possible future directions, while Section IX concludes the paper by summarizing the main findings.

II. Related Work

Over the last years, both research and industry have proposed neuromorphic processors that leverage the event-driven nature of SNNs, achieving remarkable energy efficiency.

Table I provides a summary of some of the most significant contributions from academia and industry.

Among the most prominent industrial efforts are Intel's Loihi [1] and IBM's TrueNorth [4], two highly parallel, digital neuromorphic chips specifically designed for SNN processing. Loihi, fabricated in a 14 nm Intel process, exhibits high parallelism and low power consumption, making it suitable for energy-efficient SNN execution. TrueNorth, implemented in a 28 nm low-power CMOS process, adopts a massively parallel architecture with a large number of simple spiking cores operating asynchronously.

These platforms have demonstrated high energy efficiency across several tasks, with Loihi used for gesture classification based on Electromyography (EMG) signals [21], [22] and for Dynamic Vision Sensor (DVS)-based gesture recognition [23], while TrueNorth has been employed in Electroencephalography (EEG) data analysis [24]. However, access to these systems is highly restricted, as it is often limited to remote servers or specific research agreements, which poses a significant barrier to broader adoption, particularly in contexts that require low-cost, open, and easily replicable edge AI solutions.

Several neuromorphic accelerators have been developed in academic research, such as those presented in [2], [3], [16]–[20], achieving excellent performance in terms of both energy efficiency and throughput. SNE [2] and SNPU [17] focus on the efficient processing of event-driven data, achieving low energy consumption of 0.221 pJ/SOP and 0.35 pJ/SOP, respectively, through architectures optimized for input sparsity. In contrast, SpiNNaker [3] does not prioritize energy efficiency but rather aims to investigate how information is represented and processed in the brain, enabling large-scale, real-time simulations of spiking neural networks via a massively parallel, event-driven architecture. MorphIC [16], ODIN [20], and ReckOn [18] instead emphasize on-chip learning capabilities and low-power operation, making them ideal for embedded and IoT-oriented applications. Finally, Unicorn [19] addresses the scalability of SNNs by supporting unbounded fan-out and flexible fan-in connectivity, overcoming limitations of previous architectures in handling increasingly large and complex neural topologies. While these designs achieve excellent results, their development relies on commercial tools and PDKs, which constitute a barrier to their reproduction and broader adoption in new use cases.

A more accessible path toward the widespread adoption of neuromorphic hardware is the implementation of SNN accelerators on FPGAs. Notable works such as [5] and [8] have explored FPGA-based architectures to overcome the limitations of ASIC solutions.

Presented in [5], Spiker+ is a framework for generating FPGA-based neuromorphic accelerators designed for efficient SNN inference at the edge. Synthesized on a Xilinx Artix-7 FPGA, it supports configurable LIF neuron models and customizable network topologies. Spiker+ demonstrates competitive performance on datasets such as MNIST, SHD, and AudioMNIST, with low power consumption and limited hardware resource usage, making it suitable for edge scenarios with strict power and area constraints.

In [8], a low-power FPGA is employed to achieve even greater energy efficiency, making the system particularly well-suited for edge applications involving sensor data analysis. The architecture integrates a RISC-V processor, a reconfigurable

[1]https://github.com/IHP-GmbH/IHP-Open-PDK

TABLE I
RELATED WORKS

Work	Platform	Process [nm]	Area $[mm^2]$	F_{clk} [MHz]	GSOP/s	$P_{Inference}$ [mW]	P_{Idle} [mW]	Energy/SOP [pJ]	TSOP/s/W
This Work	ASIC	130	6.8	125	2	73	0.17	36.5	0.027
SYNtzulu [8]	FPGA	40	-	22	0.18	12.4	1.23	68	0.015
Spiker+ [5]	FPGA	28	-	100	-	180	-	1370	-
OpenSpike [15]	ASIC	130	33	20	-	225	-	-	0,056
Spinnaker [3]	ASIC	130	19	200	-	653	-	7500	-
MorphIC [16]	ASIC	65	3.5	55	0.11	-	2.27	51	-
TrueNorth [4]	ASIC	28	430	Async	58	65	-	-	0.046
SNPU [17]	ASIC	28	6.3	200	-	-	-	0.35	36.5
ReckOn [18]	ASIC	28	0.86	115	-	-	-	12.8	-
UNICORN [19]	ASIC	28	500	1000	3600	85000	-	-	0.424
ODIN [20]	ASIC	28	0.086	75	-	-	-	12.7	-
SNE [2]	ASIC	22	-	400	51.2	11.29	-	0.221	4.54
Loihi [1]	ASIC	14	60	-	-	-	-	23	-

module that converts continuous sensor signals into spike trains depending on the use case, and a dedicated accelerator for SNNs. The accelerator is specifically designed to execute small-scale feedforward SNNs. Despite the limited network size, the system demonstrates near state-of-the-art accuracy across multiple use cases [9]–[11], while maintaining real-time inference capabilities and power consumption levels compatible with near-sensor edge processing.

While FPGA-based implementations offer notable advantages, including increased accessibility and reduced development costs, they fail to achieve the energy efficiency of ASIC solutions in real-time sensor data processing. In such use cases, most energy savings come from avoiding power wastage during periods of stationary input. However, processing platforms like FPGAs introduce a significant static power component that is difficult to mitigate.

An alternative pathway is provided by the rapidly expanding ecosystem of open-source EDA tools and open PDKs, which allows the design of fully digital ASICs without relying on commercial toolchains or proprietary technologies. Tools such as Yosys, OpenROAD, and KLayout, together with design flows like OpenROAD flow-scripts, OpenLANE and Silicon-Compiler, support the entire physical design process from Register-Transfer Level (RTL) to GDSII. Although currently limited to legacy technology nodes, including Sky130 (provided by SkyWater), GF180 (from GlobalFoundries), and IHP SG13G2, these platforms represent a major step forward in democratizing access to custom, low-cost, and energy-efficient neuromorphic hardware development.

Of particular relevance to this work is [15], which presents an SNN accelerator implemented using the OpenLANE open-source design flow and the Sky130 technology node. The accelerator is designed to process DVS (Dynamic Vision Sensor) images and achieves remarkable performance by exploiting binary weight representation and a highly parallel processing architecture. Although fabricated using an older 130 nm technology, it still delivers an energy efficiency of 0.056 Tera Operations per Second per Watt (TOPS/W) , comparable to that of some accelerators built with more advanced 28 nm nodes.

Our work extends [25], where a system-on-chip integrating a RISC-V core and an SNN accelerator was implemented using open-source EDA tools and an open-source PDK. In this paper, we present an improved design that reduces power consumption while enhancing throughput. Specifically, we extend the original dual-core architecture into a quad-core version, leveraging its inherent scalability and the customizability offered by ASIC design, thereby achieving a higher throughput of 2 GSOP/s and an improved energy efficiency of 36.50 pJ per operation. Additionally, we provide a more detailed comparison between the ASIC and FPGA implementations to highlight trade-offs in terms of performance, energy, and design accessibility.

While most existing neuromorphic accelerators are designed to support large-scale SNNs and are often optimized for processing data from DVS cameras, our system targets a different design space. It focuses on the efficient execution of small neural networks tailored for data from conventional low-bandwidth sensors. The proposed accelerator provides a more accessible and lightweight solution, well-suited for scenarios where resource constraints, simplicity, and reproducibility are essential.

III. SYSTEM ARCHITECTURE

The architecture presented in this work [2] builds upon the open-source[3] neuromorphic processor introduced in [8], originally designed for sensor data processing on resource-constrained FPGA platforms. Figure 1 shows the system block diagram. Highlighted in green are the three main functional blocks of the system:

- a RISC-V softcore processor responsible for system management and control.
- a delta-modulation-based encoding module that converts continuous signals into spike trains.
- an SNN accelerator that processes the spikes generated by the encoding module.

To better exploit input sparsity and take advantage of idle periods, the architecture incorporates a clock-gating mechanism that allows each module to suspend its clock signal when inactive. The RISC-V processor orchestrates this mechanism using a timer operating at 10 kHz, which periodically triggers wake-up events to check for pending processing tasks. This approach leads to a substantial reduction in energy consumption, making the system more suitable for energy-constrained

[2]https://github.com/EOLAB-2025/SYNtzulA
[3]https://github.com/gianlucaleone/SYNtzulu

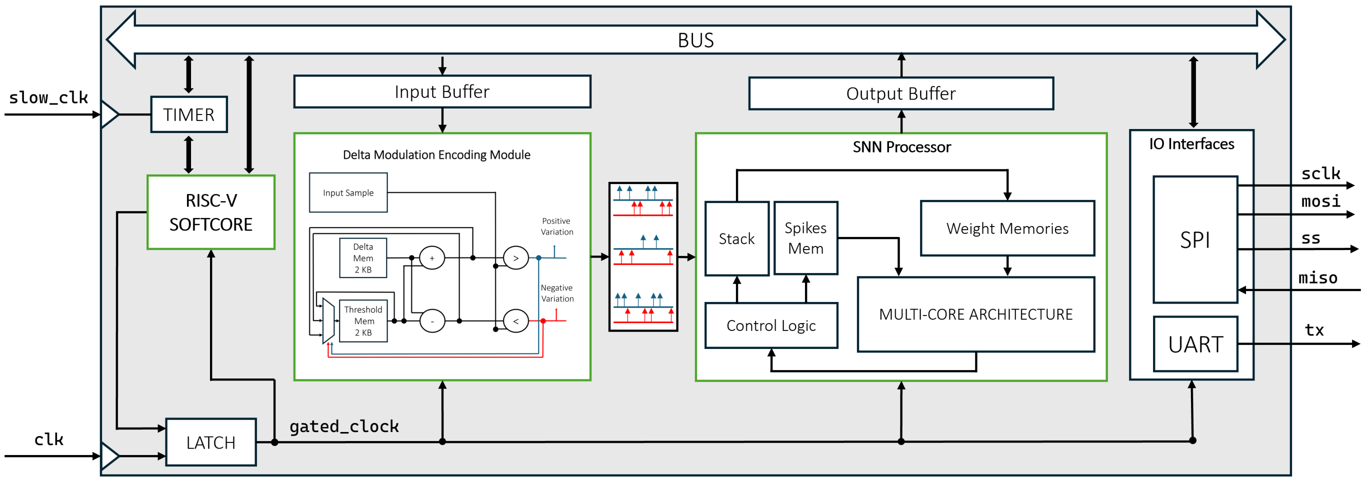

Fig. 1. Architecture Overview: SYNtzulA is composed of a timer, responsible for waking up the system after clock gating, a RISC-V softcore that manages housekeeping tasks and power management, and the accelerator that includes an encoding module for converting continuous signals into spike trains and an SNN processor for executing spiking neural network inference. Finally, the system integrates SPI and UART interfaces to handle input/output communications.

applications. The system operates with two input clock signals, where the main clock drives the entire system and the secondary 10 kHz clock is used for the timer. Finally, the architecture can communicate with external devices through Serial Peripheral Interface (SPI) and Universal Asynchronous Receiver–Transmitter (UART) interfaces.

A. *RISC-V softcore*

To improve system flexibility, the architecture includes a RISC-V softcore processor, specifically the SErial RISC-V (SERV). This core manages the loading of synaptic weights received via the SPI interface, storing them in the accelerator's weight memory. It also oversees the reception of input data and handles power management tasks, activating low-power modes through clock gating according to the system's workload.
In this design, instruction and data share a single 8 kB memory block, allowing for more efficient resource usage. The register file is implemented with flip-flops, minimizing latency and simplifying the overall design.

B. *Encoding Module*

The encoding module converts continuous input signals into spike trains, which serve as input to the SNN processor. In this work, the adopted algorithm is based on delta modulation, one of the most widely used methods for spike encoding.
The algorithm takes as input a continuous signal and generates two spike trains as output: one corresponding to the positive variation of the signal and the other to the negative variation. The encoding process employs a dynamic threshold to track variations in the input signal. If the signal exceeds the threshold plus a fixed channel-specific constant, a spike is generated on the positive channel, and the threshold is updated by adding the same constant to its current value. Conversely, if the signal falls below the threshold minus the constant, a spike is generated on the negative channel, and the threshold is updated by subtracting the constant. No spikes are generated as long as the signal remains within this range.

The hardware implementation of the encoder operates in a time-multiplexed manner, effectively reducing the required silicon area. It integrates two 1 KB Static Random-Access Memory (SRAM) blocks to store the threshold and the constant associated with each input channel, along with an additional 1 KB SRAM accessible by the RISC-V processor for buffering the incoming input samples.
In addition, two comparators are used to identify spike conditions, along with the necessary wiring and control circuitry.

C. *SNN processor*

This accelerator is based on Leaky Integrate-and-Fire (LIF) neurons, which are described by the equation in Eq. 1:

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

In this model, 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.
The SNN processor is designed to execute fully connected layers of LIF neurons using fixed-point arithmetic. Weights are represented with 8-bit precision, while membrane potentials are stored as 16-bit values. To exploit the inherent sparsity of SNN algorithms, the synaptic current computation is performed in an event-driven manner rather than sequentially iterating through all neuron inputs. This is enabled by a stack that stores the addresses of spike groups (in sets of four) that contain at least one active spike. This approach limits memory accesses to only those weight groups that are relevant, thereby reducing energy consumption. Grouping spikes in sets of four strikes a balance between maximizing sparsity exploitation (which improves with smaller groups) and minimizing the size of the stack memory, which becomes smaller as the group size increases.

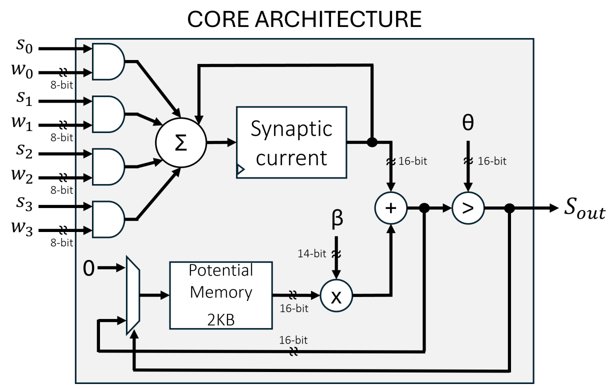

Fig. 2. Core architecture of the system. Each core includes a 4-way SIMD adder for synaptic current computation, a local memory for storing the membrane potential, a multiplier that applies a decay factor to the membrane potential, an adder that sums the decayed potential with the incoming synaptic current, and a comparator that checks whether the updated potential exceeds a predefined threshold, triggering a spike if the condition is met.

The implemented accelerator features a multi-core architecture, with each core independently computing the arithmetic for a single neuron. As illustrated in Figure 2 a single core is composed of a 4-way Single Instruction Multiple Data (SIMD) adder for computing synaptic currents, a local memory for storing the neuron's membrane potential, a multiplier that applies a decay factor to the membrane potential, an adder that sums the decayed potential with the incoming synaptic current, and a comparator that checks whether the updated potential exceeds a predefined threshold, triggering a spike if the condition is met. Weights are stored across multiple SRAM blocks, with a total capacity of up to 32,768 entries. Each core is equipped with a dedicated 2 KB memory for membrane potentials, capable of holding 1024 values. Finally, memories for storing emitted spikes and the stack are implemented using flip-flops, offering an area-efficient solution for small-capacity storage elements. The network size is constrained by the on-chip memory capacity since all weights and activations are stored locally. Weights are initialized once and remain constant throughout inference thus no external memory writes are required.

IV. RTL-GDSII FLOW

We hardened the system using the OpenROAD Flow Scripts (ORFS), a reference RTL-to-GDSII design flow developed and maintained by the OpenROAD project. ORFS offers a complete, fully open-source [4] backend implementation pipeline that includes synthesis, floorplanning, placement, clock tree synthesis, routing, and final sign-off. It supports various open and commercial PDKs and integrates example platforms and designs to streamline development and benchmarking.
For our implementation, we used the IHP SG13G2 PDK, a 130 nm BiCMOS technology.
The synthesis step was performed using Yosys, a leading open-source [5] synthesis engine widely adopted in open-source EDA flows. Following synthesis, the OpenROAD [6] tool handled all backend stages, including floorplanning, placement, clock tree synthesis, and routing. Finally, the complete layout was generated using KLayout [7], which provided the final GDSII output.
The most challenging step in the design flow was the floorplanning phase, mainly due to the presence of several memory blocks within the architecture. In our design, the automatic placement of these memories often led to inefficient area utilization, with memory macros occupying more space than necessary. To address this issue, we opted for manual memory placement, which allowed us to better control the layout and reduce area overhead.
However, this manual step introduced further complexity. Depending on how the memories were arranged, we encountered varying degrees of routing congestion, requiring several layout iterations to identify a configuration that ensured both routability and timing closure. In addition, the relative placement of memory blocks and I/O pads strongly influenced the quality of the placement of standard cells. In some cases, poor placement caused standard cells to be distributed too far apart, resulting in long interconnect paths during routing. These long paths introduced significant parasitic resistance and capacitance, ultimately reducing the maximum achievable clock frequency of the design. Therefore, careful floorplanning and iterative refinement were crucial to obtaining a timing-optimized and area-efficient physical implementation.

V. FPGA TO ASIC EVOLUTION

This section presents the key design choices that progressively shaped the final ASIC implementation. Beginning with an initial FPGA-based prototype, we developed a first version of the ASIC by adapting the architecture to the chosen PDK. In this step, FPGA-specific components such as block RAMs (BRAMs), single port RAMs (SPRAMs), and Digital Signal Processor (DSP) blocks were replaced with equivalent elements available in the PDK. After completing this initial version, we introduced a set of modifications that leverage the increased flexibility of the ASIC platform to improve the system's energy efficiency.

A. Leveraging the Flexibility of the ASIC Design

The architecture is designed to be easily scalable, allowing the number of cores to be increased to boost throughput. At the same time, resource usage grows sub-linearly with the number of cores, as control logic is shared across them rather than duplicated within each core. This shared-control approach enables efficient scaling. However, this flexibility could not be fully exploited in the original low-power FPGA implementation, due to limited hardware resources such as memory blocks and flip-flops, which constrained the number of instantiable cores. In contrast, the ASIC version overcomes these limitations, as the flexibility of custom silicon design enables the integration of additional cores without being restricted

[4]https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
[5]https://github.com/YosysHQ/yosys
[6]https://github.com/the-openroad-project
[7]https://www.klayout.de/

by fixed resource availability. This allows the architecture to achieve higher throughput with only modest increases in area and power, significantly reducing the energy per synaptic operation. To leverage this advantage, we developed not only a dual-core version of the accelerator, similar to the one implemented in [8], but also a new quad-core version. This effectively doubles the throughput and reduces the energy dissipated per synaptic operation.

B. Small Memories as Register Files

One of the early challenges in the ASIC implementation concerned the spike memory and the stack.
Implementing these using the SRAM available in the PDK wouldn't have been efficient. Each of the four required memories would only need about 256 bits, but the smallest SRAM macro in the PDK is 1 Kbit, leading to wasted, unused area. Consequently, this would have necessitated four separate 1 Kbit memory blocks, each occupying approximately 0.05 mm^2, for a total area of 0.20 mm^2.
To optimize area usage, these memories were instead implemented using flip-flops. This approach enabled an 86% reduction in area compared to the PDK SRAM-based solution, while also contributing to lower leakage power by avoiding unnecessary memory blocks.

C. Weight memories

A significant portion of the power consumption during inference derives from weight memories. In the baseline architecture, each processor is equipped with a dedicated weight memory, with data being read 32 bits at a time. All weight memories are active simultaneously, as weight fetch operations are performed in parallel by the cores. Since the PDK does not provide memory macros with 32-bit read ports, we instead had to use the available macros with 64-bit read ports. Namely, two 16 KB memories were used for the dual-core architecture, while the quad-core architecture employs four 8 KB memories. To take advantage of 64-bit width, we shared the weight memories between the cores and reorganized the weight storage layout so that each memory macro provides weights to two cores in one access. As a result, memories are accessed alternatively: in the dual-core architecture, only one of the two memories is active at each cycle (two out of four in the quad-core configuration). In this way, we can halve the number of simultaneously active memory blocks and stand-by the unused ones, reducing the power consumption of the weight memories the overall power dissipated during inference. To quantify the available bandwidth, during inference each core fetches 32 bits of weight data per clock cycle, corresponding to four 8-bit weights. In the dual-core configuration, 64 bits are read per cycle from a single 64-bit-wide memory, resulting in a bandwidth of 1 GB/s at 125 MHz. In the quad-core configuration, two 64-bit memories are accessed in parallel, providing a total bandwidth of 128 bits per cycle, i.e., 2 GB/s under the same operating frequency.

D. Dual Port Memories for Membrane Potential and Encoding

The FPGA implementation leveraged true dual-port BRAMs, enabling simultaneous read and write operations at different addresses in one clock cycle. This capability was essential for efficiently implementing the neuron membrane potential memories and the dynamic threshold memory used in the encoding process.
Since the target PDK does not provide dual-port memory blocks, an alternative approach was required. The baseline solution, presented in [25], replaces each dual-port memory by combining two single-port memory blocks used as a ping-pong buffer.
However, the use of additional memory blocks not only increases the number of macros, but also complicates physical routing. As a result, more whitespace must be reserved to facilitate signal routing between the memories and the processing logic, raising silicon area and static power consumption.
As an optimization, we exploited 64-bit macro width, allowing four neuron membrane potentials to be stored per memory word. This approach increases both read and write bandwidth by a factor of four. The potentials read from memory and potentials to be updated during writing, are temporarily stored in flip-flop buffers, and read/write occurs only when all four values are consumed/ready. This method reduces memory access rate by a factor of four, avoiding contention for conflicting read/write requests.

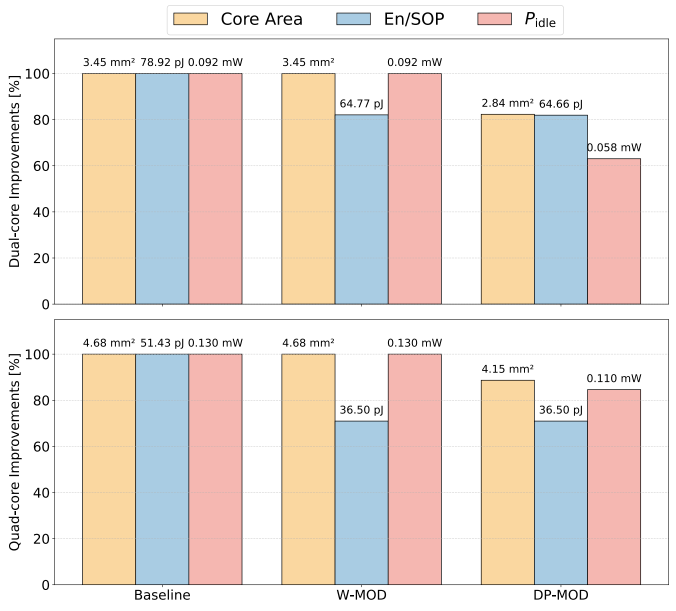

Fig. 3. Improvements achieved across the different design versions, expressed in terms of core area, energy per synaptic operation, and idle power consumption. The bar chart highlights the progressive optimization steps that led to the final ASIC implementation.

E. Ablation study

To assess the impact of the previously mentioned design choices, we compared results obtained at different adaptation stages. As a baseline implementation, we consider the most direct replacement of weight memories, implemented with SPRAMs on the FPGA, with SRAM blocks or flip-flops available in the ASIC, as presented in [25]. Using this approach, we developed the baseline implementation of both the dual-core and quad-core versions. In Figure 3, such baseline is shown by the leftmost bar. The bar labeled W-MOD illustrates the improvement achieved by optimizing the weight memories. Specifically, this reduces power consumption during inference, with an 18% decrease for the dual-core version and 29% for the quad-core version, while idle power and area remain unchanged. The second modification involved the new implementation of the dual-port memories, indicated in the figure as DP-MOD. This change made it possible to reduce the number of memory blocks, resulting in an area reduction of 18% in the dual-core case and 11% in the quad-core case. Moreover, it led to a decrease in static power consumption of 37% for the dual-core version and 15% for the quad-core version.

VI. Results

This section presents the main results of the ASIC implementation, focusing on area occupation, timing performance, and power consumption. Table II summarizes the general characteristics of the chips.

A. Area

Figure 4 shows the final layout of the two systems, with the dual-core version displayed at the top and the quad-core version at the bottom. In the dual-core version, the two large memory blocks at the top correspond to the weight memories, with smaller blocks on the left dedicated to membrane potentials, the encoding module, and the buffers used for reading from and writing to the accelerator, while the RISC-V core memory is located in the bottom-right corner. In the quad-core version, the layout shifts slightly: membrane potential memories occupy the top, weight memories are positioned in the center, encoding and buffer blocks are found at the bottom left, and the RISC-V core memory remains at the bottom right. The red regions highlight the standard cells that implement the system logic, while the I/O pins are visible along the perimeter of the layout.

The chip occupies a total area including I/O pads of approximately 5.09 mm^2 for the dual-core implementation and 6.79 mm^2 for the quad-core version. The relative sizes of the two implementations highlight a key aspect of the system's scalability: doubling the number of cores does not lead to a proportional increase in silicon area. As previously discussed, the architectural design allows for more efficient resource utilization, enabling performance to scale more effectively than the associated hardware overhead.

Figure 5 shows the area occupation percentages for the main modules in both the dual-core and quad-core versions. Excluding the I/O pads, which account for 44% of the area in the dual-core case and 39% in the quad-core, the majority of the remaining area is occupied by memory blocks, accounting for 34% and 30%, respectively. The rest of the area is used by logic circuitry and unused whitespace.

B. Timing

Timing analysis was performed using OpenSTA[8]. In both implementations, the critical path is associated with the control

[8]https://github.com/The-OpenROAD-Project/OpenSTA/tree/master

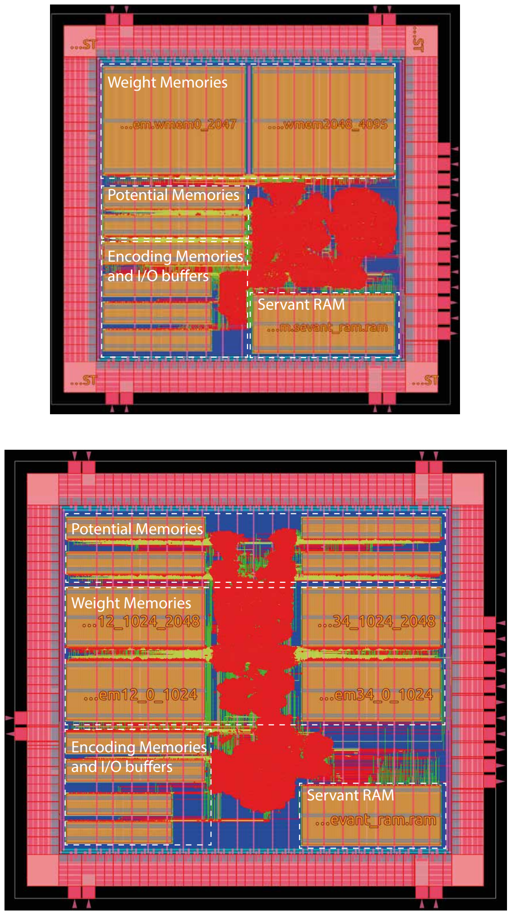

Fig. 4. Final layout of the dual-core accelerator (top) and the quad-core accelerator (bottom), visualized using the OpenROAD GUI.

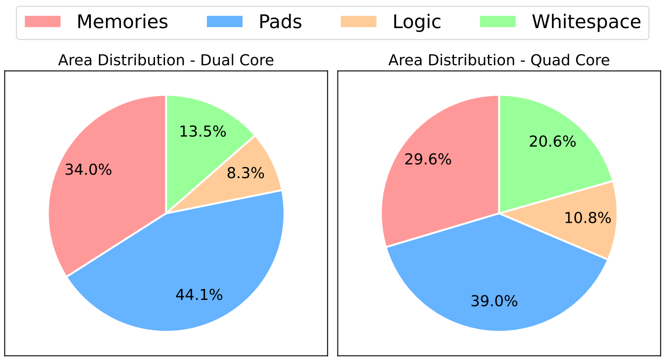

Fig. 5. Breakdown of area occupation across system components, including memories, I/O pads, logic standard cells, and whitespace.

logic that drives the weight memory address. Although the exact critical path differs between the dual-core and quad-core architectures, it originates from a similar functional unit. The critical path delay is 5.98 ns in the dual-core case and 6.83 ns in the quad-core case.

Both designs were synthesized with a target clock frequency of 125 MHz. Since each core performs four synaptic additions per clock cycle, the throughput is calculated as:

$$\text{Throughput} = N_{\text{cores}} \times f_{clk} \times 4\text{SOP} \tag{2}$$

where N_{cores} is the number of accelerator cores and f_{clk} is the clock frequency. At the maximum frequency of 125 MHz, this results in a peak throughput of 1 GSOP/s for the dual-core architecture and 2 GSOP/s for the quad-core architecture.

C. Power

To estimate power consumption, we simulated the post-layout netlist and used the resulting Value Change Dump (VCD) file to perform power analysis. Power analysis was performed using two different tools: OpenSTA and Cadence Genus. We adopted this two step approach for two main reasons. First, to cross-validate the power measurements and ensure consistency across different analysis methodologies. Second, because we aimed to assess how effectively the system exploits the event-driven nature of the algorithm, which requires estimating power consumption over different time intervals, specifically distinguishing between spike-processing phases and idle periods. Since OpenSTA does not support time-based power analysis, Cadence Genus was used to obtain accurate power profiles across different time intervals.
Power measurements were conducted at various clock frequencies in both operating phases. In the dual-core implementation, inference power increases at a rate of 0.517 mW/MHz, while in the quad-core design it scales at 0.584 mW/MHz. Idle power consists of a constant leakage component and a frequency-dependent term, which increases at approximately 545 nW/MHz in the dual-core case and 590 nW/MHz in the quad-core case.
Although the power dissipated during the inference phase is higher in the quad-core implementation, the throughput is doubled compared to the dual-core version, while power increases by only 13.3%. This results in a higher energy efficiency for the quad-core design.
To better compare the energy efficiency of the two accelerators, we adopt a commonly used metric in the state of the art: the energy dissipated per synaptic operation.
The energy per synaptic operation (En/SOP) was computed as:

$$\text{En/SOP} = \frac{\text{P}_{\text{Inference}}}{N_{\text{cores}} \times f_{clk} \times 4\text{SOP}} \tag{3}$$

The measured values are 64.66 pJ and 36.5 pJ for the dual-core and quad-core implementations, respectively. These results confirm the higher energy efficiency of the quad-core design, despite its higher absolute power consumption, and further support the scalability of the proposed architecture.
Figure 6 shows the percentage of power dissipated by the various system components during inference. In both the dual-core and quad-core cases, the majority of the power is consumed by the memory blocks. In the dual-core design, the second most power-hungry component is the clock tree, whereas in the quad-core version it is the sequential logic. This higher percentage in the quad-core is due to the presence of twice as many dual-port memories, which significantly increases the number of flip-flops used to implement them.

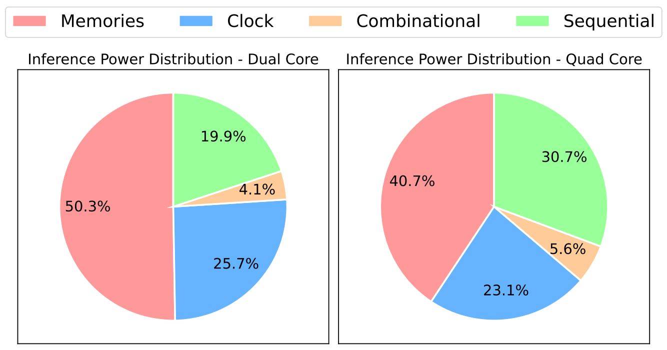

Fig. 6. Breakdown of power consumption across system components during inference: memories, clock tree, sequential elements, and combinational logic.

TABLE II
ASIC REPORT

	Dual Core	Quad Core
Die Area	2230 x 2285 μm	2735 x 2485 μm
Core Area	1660 x 1715 μm	2165 x 1915 μm
F_{MAX}	125MHz	125MHz
Performance	1 GSOP/s	2 GSOP/s
$\text{P}_{\text{Inference}}$	0.517 mW/MHz	0.584 mW/MHz
$\text{P}_{\text{Leakage}}$	0.046 mW	0.097 mW
P_{Idle}	$\text{P}_{\text{Leakage}}$ + 545 nW/MHz	$\text{P}_{\text{Leakage}}$ + 590 nW/MHz
En/SOP	64.66 pJ	36.50 pJ

VII. Discussion

Since the objective of our work is to provide an alternative that contributes to democratizing access to neuromorphic hardware, we performed a comparison with the original system implemented on a low-power FPGA, which currently represents one of the most accessible platforms for this class of applications, as well as with another SNN accelerator developed using a similar approach based on open-source EDA tools and an open-source PDK. While the efficiency gap between open-source and commercial EDA tools remains an open issue, we expect that the on-hardware performance of our design could further improve with more sophisticated toolchains. To preliminarily assess this potential, we compared the place-and-route results of the LIF neuron module obtained using both the OpenROAD-flow scripts and commercial tools (Cadence Genus and Innovus), while maintaining the same open-source PDK (IHP-SG13G2). The commercial implementation achieved a 34% improvement in maximum operating frequency, a reduction of 78% in dynamic power and a 4.6% decrease in total area. These results, although limited to a single block, indicate that our overall approach could significantly benefit from future advances in open-source EDA frameworks, without compromising its accessibility-oriented objectives.

A. FPGA implementation comparison

Regarding the original architecture presented in [8], which targets an FPGA platform, our ASIC implementation provides several advantages over the original design. Unlike FPGA-based solutions, which rely on a rigid organization of memory

TABLE III
COMPARISON WITH FPGA IMPLEMENTATION

Use case	Dataset	Synapses	Sparsity	Work	Platform	Cores	T_{Inf} [μs]	T_{Idle} [μs]	En_{Inf} [nJ]	En_{Idle} [nJ]	En_{Tot} [nJ]
sEMG	[26]	23,360	77 %	[9]	FPGA	2	30	102	368	132	500
Gesture Classification				This Work	ASIC	2	30	102	347	6	353
				This Work	ASIC	4	15	51	196	6	202
iEEG	[27]	22,848	73 %	[10]	FPGA	2	34	95	418	122	540
Neural Decoding				This Work	ASIC	2	34	95	394	6	400
				This Work	ASIC	4	17	47	223	5	228
ECG	[28] [29]	17,088	85 %	[11]	FPGA	2	14	82	175	107	282
Arrhythmia detection				This Work	ASIC	2	14	82	166	5	171
				This Work	ASIC	4	7	41	94	4	98

macros, the ASIC implementation allows for greater flexibility in memory placement and sizing. This enables optimized area utilization, the ability to scale the number of memory blocks according to application requirements, and strategic placement to facilitate routing and meet timing constraints. As discussed in Section V, this flexibility has been leveraged to improve both the system's energy efficiency and performance, resulting in higher throughput and reduced energy consumption per synaptic operation compared to the FPGA-based implementation.

Moreover, the absence of reconfigurable logic eliminates the timing overheads typically associated with FPGA architectures, enabling higher maximum clock frequencies compared to the FPGA-based implementation.

Finally, one of the most important advantages of the ASIC implementation is its significantly lower idle power consumption, which enables more effective exploitation of the event-driven nature of the algorithm. This feature is crucial for fully leveraging the inherent sparsity of spiking neural networks, as it ensures that energy is not wasted during periods of inactivity. However, a meaningful comparison in terms of overall energy efficiency must also take into account the sparsity of the algorithm, which is inherently dependent on the specific neural network model. For this reason, performance evaluation must be carried out on a per-use-case basis. We selected three works [9] [10] [11] where the FPGA-based system was previously evaluated and demonstrated state-of-the-art accuracy across three different biosignal analysis use cases. In all three studies, a fully connected SNN based on LIF neurons was employed. Based on the reported model size and sparsity, we compared our ASIC implementation in terms of inference latency and energy consumption. In [9], an SNN composed of 269 neurons and approximately 23K parameters was used to decode surface Electromyography (sEMG) signals for gesture classification. The network was trained and evaluated on the [26] dataset, achieving a sparsity of 77% and an accuracy of 83.17% in the classification of twelve different finger gestures. In [10], an SNN composed of 261 neurons and approximately 23K parameters was used to decode intracranial Electroencephalography (iEEG) signals for the estimation of the hand kinematics and kinetics of a monkey. The network was trained and evaluated on dataset [27], achieving an accuracy, measured as the Coefficient of Determination (R^2) between the decoded variable and the ground truth, of 0.66 on average for hand velocity decoding and 0.76 for hand force decoding, with a sparsity of 73%. Finally, in [11], Electrocardiography (ECG) signals were analyzed for arrhythmia detection using an SNN composed of 261 neurons and approximately 17K parameters. The network was trained and evaluated on dataset [28] [29], and it classified the samples into the five most critical groups for arrhythmia recognition: N (non-ectopic beats), S (supraventricular ectopic beats), V (ventricular ectopic beats), F (fusion beats), and Q (unclassifiable beats), achieving an overall accuracy of 98.4%, which is consistent with the state of the art, and a sparsity of 85%.

Table III reports the comparison between the ASIC implementation and the FPGA-based system. The left side of the table lists the main characteristics of the model, including the dataset used for training and testing, the model size expressed as the total number of synapses, and the sparsity. The sparsity values reported for each of the three models are average values obtained by monitoring their execution over the entire test set. To evaluate the variability of sparsity throughout the inference process on the test dataset, as an example, we additionally report the maximum, minimum, and standard deviation values obtained by the model [9], which are 81%, 67%, and ±3%, respectively. The right side presents the comparison between the ASIC and FPGA implementations. To ensure a fair comparison, both implementations are evaluated at the same operating frequency of 22 MHz. The table presents the total energy consumed per inference En_{Tot}, along with the metrics used for its calculation. In particular the inference time T_{Inf} defined as the time required to process active synapses, calculated as:

$$T_{\text{inf}} = \frac{(1 - \text{Sparsity}) \times \text{Synapses}}{N_{\text{cores}} \times f_{clk} \times 4\text{SOP}} \tag{4}$$

where the numerator represents the average number of active synapses during inference, and the denominator corresponds to the throughput of the accelerator. The idle time, T_{Idle}, defined as the period during which inactive synapses are skipped, is calculated as:

$$T_{\text{Idle}} = \frac{\text{Synapses}}{N_{\text{cores}} \times f_{\text{clk}} \times 4\text{SOP}} - T_{\text{Inf}} \tag{5}$$

In this expression, the first term represents the time that would be required to process all synapses if no sparsity were present, while the second term subtracts the actual time needed to perform the inference. Finally the energy dissipated during the inference phase En_{Inf} and during the idle phase En_{Idle}.

The comparison between the dual-core ASIC and the FPGA implementation underscores the ASIC's superior ability to exploit spike sparsity. Thanks to its much lower idle-state power, the ASIC validates our initial assumption that sparsity-oriented processing benefits more from custom silicon, which does not incur the substantial quiescent power typical of FPGA fabrics. Remarkably, even though the FPGA is manufactured in a more advanced 40 nm process and the ASIC in a 130 nm node, the energy consumed during the active inference phase is essentially the same for both solutions, whereas the ASIC's idle energy is almost negligible. In contrast, for the FPGA, idle energy is reduced but still significant when compared to inference energy. As a result, the ASIC implementation consistently achieves lower total energy consumption across all three use cases.

Moreover, as shown by the comparison between the quad-core and dual-core versions, scalability further improves efficiency. Doubling the number of cores results in a proportional increase in throughput, while the energy per inference is reduced. Since idle power remains virtually negligible, scaling up the architecture does not lead to a significant energy overhead. As a result, the quad-core ASIC not only improves performance but also achieves less than half the energy dissipation of the equivalent dual-core FPGA implementation across all three use cases.

B. Comparison with Open-Source Accelerator

To the best of our knowledge, the work presented in [15] is the only other accelerator for SNNs developed using an approach similar to ours, based on open-source EDA tools and an open-source PDK. Their design also targets the 130 nm technology node, specifically the Sky130 PDK, and is tailored for processing data from Dynamic Vision Sensors (DVS).

The accelerator occupies a total area of 33 mm^2, which is five times larger than our design, with 21.89 mm^2 dedicated to 77 on-chip memory blocks of 2 KB each. These memory blocks store spikes, adaptive thresholds, membrane potentials, decay rates, weights, and incoming spikes. The accelerator and memory blocks operate at 20 MHz and 40 MHz, respectively, and dissipate 119 mW and 106 mW, resulting in a combined power consumption of 225 mW.

While their design consumes approximately fourteen times more power during inference compared to ours, at the same operating frequency, it achieves higher energy efficiency in terms of GOPS/W. This improvement is mainly due to a high degree of parallelism, with 1024 neurons processed simultaneously,at the cost of a significantly larger silicon area compared to our design. Additionally, the design is tailored for DVS workloads and employs binarized weights, resulting in significantly increased computational throughput. In contrast, our accelerator is intended for more general-purpose sensor signal processing, which imposes different design trade-offs.

According to the data in [30], the memory used in their design likely consumes around 1.36 mW of static power, although this is not explicitly reported in the original work. Additionally, no mechanism for power gating is described.

These aspects make a direct, one-to-one comparison challenging. However, the significantly higher power consumption in both inference and idle modes, combined with the absence of information regarding the exploitation of sparsity and the use of binary weights, indicates that their design may not be well suited for general-purpose, energy-constrained applications. In contrast, our approach prioritizes flexibility, efficient use of resources, and energy-aware design choices, making it more appropriate for a broader set of real-world scenarios.

VIII. Future Works and Limitations

Although the proposed architecture demonstrates competitive performance and energy efficiency despite being implemented in a legacy 130 nm technology, several limitations and directions for future improvement can be identified.

The first limitation concerns the technology node itself. The use of a 130 nm open PDK inherently constrains achievable operating frequency and power consumption compared to modern processes. Nonetheless, the comparison with the FPGA implementation manufactured in a more advanced 40 nm node highlights that even with older technology, open ASICs can outperform reconfigurable devices in energy efficiency. In the future, the availability of more advanced open-source PDKs could help overcome this limitation, enabling further improvements in performance and energy efficiency. As preliminary evidence of the potential improvements achievable with more advanced nodes, we synthesized the same design using a commercial 28 nm PDK and standard commercial EDA tools. The preliminary results indicate a maximum clock frequency of 1.5 GHz, with an estimated active power dissipation of 75.9 mW at this maximum frequency and a leakage power of 0.18 mW. These results demonstrate that the proposed architecture scales well when implemented in more advanced technologies.

A second limitation lies in the spike encoding strategy, which is currently fixed to a delta-modulation scheme. While this approach provides a good balance between simplicity and efficiency, different application domains could benefit from more flexible or task-specific encoding methods. Future work will explore a reconfigurable encoding module capable of adapting to the characteristics of the target input data and use case.

The current accelerator presents other two main limitations. First, it supports only fully connected layers composed of Leaky Integrate-and-Fire (LIF) neurons. This restriction limits the range of neural architectures that can be executed, preventing the implementation of more complex models such as convolutional or recurrent networks. As a result, the current design is best suited for lightweight sensor data processing tasks, such as biosignals or other one-dimensional time series. Nonetheless, extending the architecture to support convolutional layers would significantly broaden its applicability, enabling efficient processing of multi-dimensional data such as event-based vision or other spatially correlated sensor streams.

Second, the network size is constrained by the capacity of the on-chip weight memories. In the current implementation, weights are preloaded into these memories before inference, and no write operations can be performed during execution. Consequently, the total number of weights that can be used is fixed, and it is not possible to stream additional weights from external memory. Incorporating a Direct Memory Access (DMA) mechanism or a similar data-transfer interface could alleviate this limitation, allowing the execution of larger networks by dynamically loading weights during inference. Summarizing, future developments will focus on addressing these limitations by enhancing the hardware flexibility and computational capabilities of the system. In particular, future work will aim to design a more adaptable encoding architecture and extend the accelerator to support convolutional SNN layers, while also exploring design optimizations or alternative open-source technology nodes to further improve performance and energy efficiency.

IX. Conclusion

In this work, we presented SYNtzulA, an ASIC accelerator for Spiking Neural Network inference implemented using an open source 130 nm PDK and open-source EDA tools. The system achieves an energy efficiency of 36.50 pJ/SOP and a peak throughput of 2 GSOP at 125 MHz. Despite being implemented in a legacy technology node, it outperforms the FPGA-based implementation across multiple use cases, highlighting that open ASIC design represents a feasible path to democratizing access to neuromorphic hardware and providing a viable alternative to FPGA-based solutions. Finally we demonstrated that leveraging an ASIC-based approach allows us to fully exploit the event-driven nature of the algorithm, making the proposed architecture particularly well-suited for real-time processing of sensor data.

References

[1] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, H. Wang, and D. Mathaikutty, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. PP, pp. 1–1, 01 2018.

[2] A. D. Mauro, A. S. Prasad, Z. Huang, M. Spallanzani, F. Conti, and L. Benini, "Sne: an energy-proportional digital accelerator for sparse event-based convolutions," 2022. [Online]. Available: https://arxiv.org/abs/2204.10687

[3] E. Stromatias, F. Galluppi, C. Patterson, and S. Furber, "Power analysis of large-scale, real-time neural networks on spinnaker," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[4] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "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.

[5] A. Carpegna, A. Savino, and S. D. Carlo, "Spiker+: a framework for the generation of efficient spiking neural networks fpga accelerators for inference at the edge," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–15, 2024.

[6] J. Sommer, M. A. Özkan, O. Keszocze, and J. Teich, "Efficient hardware acceleration of sparsely active convolutional spiking neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3767–3778, 2022.

[7] J. Li, G. Shen, D. Zhao, Q. Zhang, and Y. Zeng, "Firefly v2: Advancing hardware support for high-performance spiking neural network with a spatiotemporal fpga accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 9, pp. 2647–2660, 2024.

[8] G. Leone, M. A. Scrugli, L. Badas, L. Martis, L. Raffo, and P. Meloni, "Syntzulu: A tiny risc-v-controlled snn processor for real-time sensor data analysis on low-power fpgas," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–12, 2024.

[9] M. Scrugli, G. Leone, P. Busia, L. Raffo, and P. Meloni, "Real-time semg processing with spiking neural networks on a low-power 5k-lut fpga," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, pp. 1–14, 01 2024.

[10] L. Martis, G. Leone, L. Raffo, and P. Meloni, "Low-power fpga-based spiking neural networks for real-time decoding of intracortical neural activity," *IEEE Sensors Journal*, vol. 24, no. 24, pp. 42 448–42 459, 2024.

[11] M. A. Scrugli, P. Busia, G. Leone, and P. Meloni, "On-fpga spiking neural networks for integrated near-sensor ecg analysis," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.

[12] P. Busia, G. Leone, A. Matticola, L. Raffo, and P. Meloni, "Wearable epilepsy seizure detection on fpga with spiking neural networks," *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–11, 2025.

[13] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.

[14] V. Degalahal and T. Tuan, "Methodology for high level estimation of fpga power consumption," in *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, vol. 1, 2005, pp. 657–660 Vol. 1.

[15] F. Modaresi, M. Guthaus, and J. K. Eshraghian, "Openspike: An openram snn accelerator," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[16] C. Frenkel, J.-D. Legat, and D. Bol, "A 65-nm 738k-synapse/mm2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[17] S. Kim, S. Kim, S. Um, S. Kim, J. Lee, and H.-J. Yoo, "Snpu: An energy-efficient spike domain deep-neural-network processor with two-step spike encoding and shift-and-accumulation unit," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 10, pp. 2812–2825, 2023.

[18] C. Frenkel and G. Indiveri, "Reckon: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[19] Z. Yang, L. Wang, Y. Wang, L. Peng, X. Chen, X. Xiao, Y. Wang, and W. Xu, "Unicorn: a multicore neuromorphic processor with flexible fan-in and unconstrained fan-out for neurons," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 943–948. [Online]. Available: https://doi.org/10.1145/3489517.3530563

[20] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol, "A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, pp. 145–158, 2019.

[21] A. Vitale, E. Donati, R. Germann, and M. Magno, "Neuromorphic edge computing for biomedical applications: Gesture classification using emg signals," *IEEE Sensors Journal*, vol. 22, no. 20, pp. 19 490–19 499, 2022.

[22] S. S. Bezugam, A. Shaban, and M. Suri, "Neuromorphic recurrent spiking neural networks for emg gesture classification and low power implementation on loihi," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[23] R. Massa, A. Marchisio, M. Martina, and M. Shafique, "An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[24] B. S. Mashford, A. Jimeno Yepes, I. Kiral-Kornek, J. Tang, and S. Harrer, "Neural-network-based analysis of eeg data using the neuromorphic truenorth chip for brain-machine interfaces," *IBM Journal of Research and Development*, vol. 61, no. 2/3, pp. 7:1–7:6, 2017.

[25] L. Martis, G. Leone, L. Raffo, and P. Meloni, "SYNtzulA: Open-source hardware for energy-efficient spiking neural network inference," in *Proceedings of the 22nd ACM International Conference on Computing Frontiers (CF Companion '25)*, Cagliari, Italy, May 2025, accepted for publication.

[26] S. Pizzolato, L. Tagliapietra, M. Cognolato, M. Reggiani, H. Müller, and M. Atzori, "Comparison of six electromyography acquisition setups on hand movement classification tasks," *PLOS ONE*, vol. 12, no. 10, pp. 1–17, 10 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0186132

[27] T. Brochier, L. Zehl, Y. Hao, M. Duret, J. Sprenger, M. Denker, S. Grün, and A. Riehle, "Massively parallel recordings in macaque motor cortex during an instructed delayed reach-to-grasp task," *Scientific data*, vol. 5, no. 1, pp. 1–23, 2018.

[28] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[29] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E.

Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. E215–20, Jun. 2000.

[30] sky130 datasheet. Accessed: June, 09, 2025. [Online]. Available: https://github.com/VLSIDA/sky130_sram_macros/blob/main/sky130_sram_2kbyte_1rw1r_32x512_8/sky130_sram_2kbyte_1rw1r_32x512_8.html

Luca Martis received the B.S. and M.S. degrees in electronics engineering from the University of Cagliari, Cagliari, Italy, in 2021 and 2023, respectively, where he is currently pursuing the Ph.D. degree in electronic and computer engineering. His research interests include the development of custom hardware systems for executing algorithms based on artificial intelligence, with a particular focus on spiking neural networks.

Gianluca Leone received the B.S. degree in electronics engineering from the University of Cagliari, Italy, in 2016, the M.S. degree in electronics engineering from Politecnico di Torino, Italy, in 2019, and the Ph.D. degree in electronics and computer engineering from the University of Cagliari in 2023. Since 2023, he has been an Assistant Professor with the University of Cagliari. He teaches integrated systems design and mixed-signal circuits and systems at the University of Cagliari. His recent work includes the development of FPGA-based systems for processing bio-signals and SNN-type workloads in real-time at the edge. His research interests include the design and optimization of digital systems.

Luigi Raffo (Member, IEEE) received the Laurea degree in electronic engineering and the Ph.D. degree in electronics and computer science from the University of Genoa, Italy, in 1989 and 1994, respectively. In 1994, he joined the Department of Electrical and Electronic Engineering, University of Cagliari, Italy, as an Assistant Professor and as an Associate Professor in 1998. Since 2006, he has been a Full Professor of electronics with the Department of Electrical and Electronic Engineering, University of Cagliari. He teaches courses on system/digital and analog electronic design and processor architectures for the courses of studies in electronic and biomedical engineering. He was a Coordinator of the project EU IST-FET-IST-2001-39266-BEST and the MADNESS EU Project (FP7/2007-2013), a Unit Coordinator of the project EU IST-FET-SHAPES-Scalable Software Hardware Architecture Platform for Embedded Systems, and a Local Coordinator of industrial projects in the field (among others: ST-Microelectronics-Extension of ST200 architecture for ARM binary compatibility and ST-Microelectronics-Network on chip). He is responsible for the cooperation programs in the field of embedded systems with several other European Universities. He was also a Local Coordinator of the ASAM (ARTEMIS-JU) and ALBA projects (national founded project) and RPCT (regional founded project).

Paolo Meloni (Member, IEEE) is currently an Associate Professor with the University of Cagliari. He teaches microcontroller-based systems and advanced embedded systems at the University of Cagliari. He is the author of a significant track of international research papers. His research activity is on the development of advanced digital systems, on the application-driven design and programming of multi-core on-chip architectures and FPGAs.