Energy-Efficient Digital Design: A Comparative Study of Event-Driven and Clock-Driven Spiking Neurons

Energy-Efficient Digital Design: A Comparative Study of Event-Driven and Clock-Driven Spiking Neurons

Filippo Marostica, Alessio Carpegna, Alessandro Savino, Stefano Di Carlo
Control and Computer Engineering Department, Politecnico di Torino, Turin, Italy
Email: {filippo.marostica, alessio.carpegna, allesandro.savino, stefano.dicarlo}@polito.com

***Abstract*—This paper presents a comprehensive evaluation of Spiking Neural Network (SNN) neuron models for hardware acceleration by comparing event-driven and clock-driven implementations. We begin our investigation in software, rapidly prototyping and testing various SNN models—based on different variants of the Leaky Integrate and Fire (LIF) neuron—across multiple datasets. This phase enables controlled performance assessment and informs design refinement. Our subsequent hardware phase, implemented on Field Programmable Gate Array (FPGA), validates the simulation findings and offers practical insights into design trade-offs. In particular, we examine how variations in input stimuli influence key performance metrics such as latency, power consumption, energy efficiency, and resource utilization. These results yield valuable guidelines for constructing energy-efficient, real-time neuromorphic systems. Overall, our work bridges software simulation and hardware realization, advancing the development of next-generation SNN accelerators.**

***Index Terms*—Spiking Neural Networks, FPGA, Event-Driven Processing, Clock-Driven Processing, Energy Efficiency**

I. Introduction

In recent years, Spiking Neural Networks (SNNs) have emerged as a promising paradigm for energy-efficient computation, particularly when implemented using dedicated neuromorphic hardware [1]. Unlike conventional neural networks that use continuous activation functions, SNNs emulate the discrete, event-based operation of biological neurons by transmitting information through spikes. This temporal coding mechanism enables SNNs to exploit the inherent sparsity of neural activity, potentially reducing energy consumption and improving processing speed for time-dependent tasks.

Within the realm of digital SNNs hardware accelerators, two primary processing methods have been explored: clock-driven and event-driven [2]. The clock-driven approach updates neuron states at each clock tick, conversely, the event-driven method updates neurons only when an input spike occurs reducing unnecessary computations and better mimics the behavior of biological neurons [3].

This work explores the trade-offs between event-driven and clock-driven processing for SNN neurons through a cross-layer analysis. We evaluate accuracy using software models and assess latency, power, energy, and resource usage via hardware implementations on Field Programmable Gate Arrays (FPGAs). By varying key architectural parameters—such as processing mode and input sparsity—we provide practical insights into the design of energy-efficient neuromorphic systems.

The paper is organized as follows: Section II discusses the neuron model and its event-driven and clock-driven implementations, Section III focuses on the software implementation and accuracy analysis of the neuron models, Section IV covers the hardware implementation on FPGA, Section V presents the results and performance analysis, finally, Section VI concludes the paper and outlines potential future work.

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union.

To foster research in this field, we are making our experimental code available as open source: https://github.com/smilies-polito/spiking-neurons-comparison.git

II. Neuron model

Recent research in neuromorphic computing emphasizes efficient hardware platforms [4], exploring optimization across multiple abstraction levels [5]. High-level optimizations focus on the network architecture and parameters [6], while lower-level optimization addresses fundamental computational elements like neuron models and connections. Our work focuses on the latter, analyzing how neuron processing methods impact performance. Neuron models have historically balanced biological accuracy and computational efficiency. Early biophysically detailed models like Hodgkin-Huxley [7] were computationally demanding, prompting simplified alternatives. Among these, the Leaky Integrate and Fire (LIF) model [8] captures essential neuronal dynamics efficiently, making it suitable for neuromorphic hardware. The next sections present the mathematical formulation of the standard and quantized LIF models for hardware acceleration.

A. Mathematical Model of the LIF Neuron

The LIF model is derived from an equivalent electrical circuit where the neuron's membrane is represented as a capacitor C in parallel with a resistor R, modeling ion leakage. The membrane potential $U_{\text{mem}}(t)$ follows the differential equation:

$$\tau \frac{dU_{\text{mem}}(t)}{dt} = -U_{\text{mem}}(t) + RI_{\text{in}}(t) \quad (where\ \tau = RC)$$

The neuron exhibits a threshold-based firing mechanism: when $U_{\text{mem}}(t)$ exceeds a predefined threshold $U_{\text{threshold}}$, the neuron emits a spike and resets its membrane potential.

B. Event Driven and Clock Driven implementation

For digital hardware implementations, the continuous-time model solution is discretized, obtaining:

$$U_{\text{mem}}[t + \alpha \cdot t] = \beta U_{\text{mem}}[t] + WX[t] - R[t]$$

Where W is the weight, $X[t]$ the input at time t, $R[t]$ the reset applied on spike, and $\beta = e^{-\frac{\alpha \cdot t}{\tau}}$ is the exponential decay coefficient.

Both clock-driven and event-driven models compute the exponential decay of the membrane potential using the decay factor β, but they differ in how and when this evaluation occurs. In the clock-driven approach, the neuron updates its membrane potential at every clock cycle by multiplying it by β. In the literature, this has been implemented either with an exact multiplier or with a shifter that approximates the multiplication by the nearest power of two [1], [9]. In contrast, the event-driven approach updates the membrane potential only when an event occurs. This requires evaluating the time elapsed between two consecutive spikes ($\Delta t = t_{i+1} - t_i$). In the absence of input spikes, the membrane potential follows an exponential decay governed by the recursive relation: $U_{\text{mem}}(t+1) = \beta U_{\text{mem}}(t)$. By iterating over two time steps where no input occurs, this extends to: $U_{\text{mem}}(t+2) = \beta^2 U_{\text{mem}}(t)$; and generalizing on n time steps between two active input spikes: $U_{\text{mem}}(t+n) = \beta^n U_{\text{mem}}(t)$.

Since Δt is variable, the computation of $\beta^{\Delta t}$ must be efficient. To achieve this, a Look-Up Table (LUT) is typically used, storing precomputed values of $\beta^{\Delta t}$ for different intervals. As in the clock-driven case, two implementations are possible: (1) storing exact values of $\beta^{\Delta t}$, functionally equivalent to using a multiplier, and (2) approximating $\beta^{\Delta t}$ with the closest power of two, enabling a hardware-efficient shifter-based implementation.

Besides their mathematical differences, event-driven neurons differ from clock-driven ones in how inputs are handled. Clock-driven neurons process inputs serially, while event-driven neurons can do so either serially or via Address Event Representation (AER), a protocol widely used in neuromorphic systems that encodes spikes as addressed events. Although AER can be adapted to clock-driven systems, its overhead makes it more efficient for event-driven designs. AER also directly interfaces with neuromorphic sensors (e.g., silicon retinas [10] or silicon cochleae [11]), reducing bandwidth by avoiding continuous data streams. Our comparative analysis examined six configurations: (i) clock-driven serial neuron with an exact multiplier, (ii) clock-driven serial neuron with a shifter, (iii) event-driven serial neuron with an exact multiplier, (iv) event-driven serial neuron with a shifter, (v) event-driven AER neuron with an exact multiplier, and (vi) event-driven AER neuron with a shifter.

III. Software Implementation

The first phase of this work focuses on analyzing the impact of the considered neuron models on the accuracy of SNNs built on top of them. While software models cannot fully replicate the exact behavior and constraints of hardware, they are valuable tools for validating the computational model, analyzing the accuracy in specific tasks (e.g., classification), and identifying potential bottlenecks.

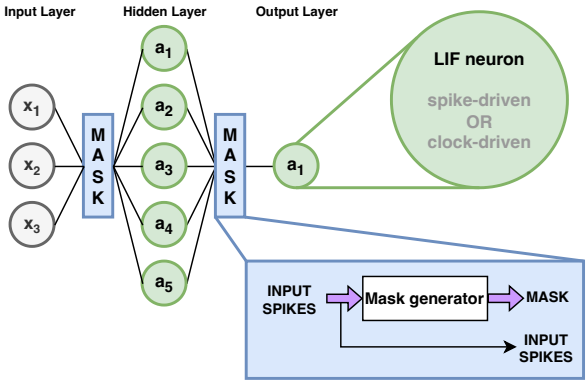

Fig. 1. High-level view of an SNN architecture incorporating a custom mask layer to track input activity an two custom neuron layers configurable to operate either in a spike-driven or clock-driven mode.

For this software-level analysis, the SnnTorch [12] Python framework was selected due to its seamless integration with PyTorch [13]. Along with the default SnnTorch's LIF neuron a custom event-driven neuron was developed by overriding the standard LIF one. For this preliminary software-level analysis, the difference in spike encoding between serial and AER is irrelevant and, therefore, not considered.

Figure 1 illustrates the custom neuron model, consisting of a configurable LIF neuron and a mask generator. The mask flags active inputs at each time step, allowing event-driven neurons to process only when needed, while ensuring consistent input patterns across both neuron types for fair comparison.

A. Sparsity Analysis on datasets

Our analysis compares three datasets reflecting different spiking-data scenarios (see Table I). The datasets are characterized by different input densities along two dimensions—temporal density (i.e., frequency of updates) and per-time step input activity (i.e., active inputs per timestep)—obtained by analyzing a subset of 1,000 elements from each dataset. These benchmarks cover three common scenarios—spiking by nature, static-to-spike conversion, and dynamic-to-spike conversion. Figure 3 illustrates their differing temporal densities, with each dataset corresponding to a distinct band. MNIST [14] comprises static images converted to spikes via Poisson rate-coding, yielding nearly 100% temporal density. N-MNIST [15] is inherently spiking data recorded by a dynamic ATIS sensor, and AudioMNIST [16] includes non-spiking audio data requiring spike conversion.

B. Experimental Validation and Results

To evaluate different neuron models, we compared their effects on inference accuracy across three datasets under identical training conditions with SnnTorch's standard (clock-driven) neuron. This analysis clarifies how approximate hardware models behave and guides their integration into full

TABLE I
CLOCK-DRIVEN VS. EVENT-DRIVEN NEURONS INFERENCE ACCURACY AND DATASET CHARACTERISTICS

	MNIST	N-MNIST	AudioMNIST
Inference Accuracy (%)			
Clock-driven (Multiplier)	99.22	96.24	96.09
Clock-driven (Shifter)	99.22	96.24	96.09
Event-driven (Multiplier)	99.22	96.24	96.09
Event-driven (Shifter)	99.22	96.23	96.09
Dataset Properties (%)			
Temporal Density	100 ± 0	93.7 ± 3.80	16.6 ± 2.8
Input Density	13.2 ± 5.1	1.6 ± 0.5	74.8 ± 12.7

networks. We specifically examine how approximated decay coefficients affect accuracy in event-driven neurons compared to clock-driven ones, focusing on minimizing errors introduced by shifter-based approximations.

In multiplier-based neurons, any value of β can be used without significant hardware or algorithmic constraints. However, in shifter-based neurons, an "optimal" β must be chosen to minimize accuracy loss. Typically, β is near 1, and to reduce quantization errors when approximating with powers of 2, we define $\beta = 1 - \beta'$, where β' is a power of 2 (i.e., $\beta' = 2^{-n}$). For instance, $\beta = 0.9325$ ($\beta' = 0.0625 = 2^{-4}$) for the AudioMNIST and MNIST datasets, and $\beta = 0.5$ ($\beta' = 0.5 = 2^{-1}$) for N-MNIST. Although this approach increases area slightly (due to extra logic), it significantly reduces the error compared with a simpler shifter-based scheme.

Despite minor inaccuracies from LUT-based approximations of $\beta^{\Delta t}$, software simulations show minimal accuracy differences between event-driven and clock-driven neurons. While sparsity affects behavior slightly—especially in sparse datasets—the shifter-based design provides a good balance between efficiency and accuracy. Future work could examine network training directly on quantized models to further improve precision. Overall, shifter-based neurons perform comparably to multiplier-based versions across various tasks and data sets.

IV. HARDWARE IMPLEMENTATION

The hardware implementation of the different neurons is designed with modularity in mind, comprising a top entity, a Datapath (DP) and a Control Unit (CU). Rather than analyzing a complete system architecture, our focus is on a fine-grained study of the architecture of a single neuron. In this design, the top entity handles communication with memory, where parameters are stored and receives spikes from external sources. The DP processes the data, including updating the membrane potential and checking firing conditions, based on control signals generated by the CU. This modular approach was designed and implemented in SystemVerilog, targeting Xilinx FPGAs, and a high-level representation of the neuron structure is shown in Figure 2. Moreover in our implementation, no Digital Signal Processor (DSP) or Block RAM (BRAM) blocks were used. Although modern FPGAs provide DSP slices, their use is not guaranteed by the synthesis tool and can vary based on tool optimizations, synthesis directives, and resource availability. To ensure a fair and generalizable comparison, we enforced the use of LUT-based arithmetic across all neuron variants. This avoids backend-specific optimizations and enables consistent post-synthesis evaluation of area and power, independent of device-specific heuristics.

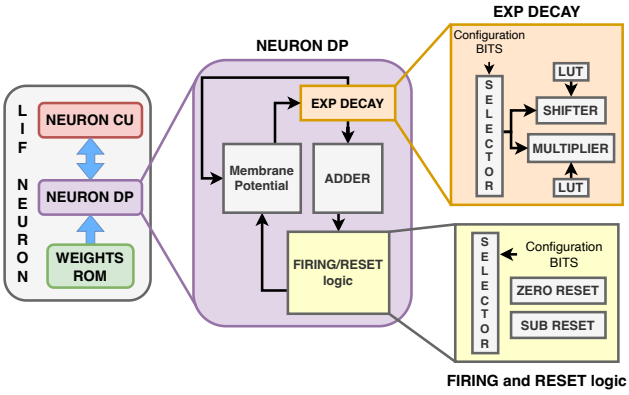

Fig. 2. High-level schematic of the hardware implementation of the LIF neuron. The top entity (left) includes a Control Unit (CU) and a Data Path (DP), along with a weights ROM for parameter storage. Within the DP (center), the membrane potential is updated by an exponential decay stage and an adder, and the neuron's firing/reset logic is managed by configurable blocks. The exponential decay can be computed via either a shifter or a multiplier (right, orange box), accessed from different LUT entries according to user-defined bits, while the firing/reset logic (right, yellow box) also adapts to configurable bits that select between different reset strategies (e.g., zero or subtract reset).

A. Clock-Driven Implementation

In the clock-driven implementation, inputs are received as a string of bits, where a '0' indicates no spike and a '1' indicates a spike. During each time step, if all inputs are zero, it means no input spikes are present, and the neuron simply updates its membrane potential. When a time step contains at least one input different from zero, the neuron processes the inputs serially, executing the mathematical calculations described in Section II-A. While pipelining the serial datapath could increase throughput by overlapping operations within a single time step, we chose a non-pipelined design to maintain consistency across all six neuron variants. This ensures a fair comparison of architectural trade-offs without introducing implementation-dependent optimizations that would affect area and power. The goal of this work is not to maximize absolute performance, but to isolate and compare the inherent efficiency of each neuron processing strategy under equivalent design conditions.

B. Event-Driven Implementation

In the serial case, when a time step consists entirely of zeros (indicating no input spikes), as for the clock-driven neuron, the neuron increments a counter to keep track of the time since the last spike. On the other hand, when the neuron is configured to operate with AER data, the system uses the information embedded into each AER string to access the memory, and a register is used to keep track of the time.

V. Simulation and Results Analysis

We performed simulations to evaluate performance metrics for both clock-driven and event-driven neuron models varying the level of sparsity. Data were generated in Serial and AER formats, with sparsity varying along the two dimensions: Input Sparsity and Temporal Sparsity. We focused on processing mode, sparsity, and decay implementation (shifter vs. multiplier) because these are among the most influential factors in hardware performance. This choice enables a targeted evaluation of trade-offs in latency, energy, and area under realistic constraints.

All designs run at 100 MHz, to align with data reported in previous publications on FPGA-specific hardware accelerators such as [17] or [18]. Because power and latency scale almost linearly with frequency, changing the clock would shift absolute values but not the relative trends across our six neuron variants. We simulate 100 time-steps and drive each neuron with 8 input channels. Latency, power and resource usage grow linearly with extra steps or channels, so varying them would not change the qualitative conclusions. A testbench capable of interfacing with both the clock-driven and event-driven neuron models was developed, allowing a direct comparison of their performance. All neurons use 6-bit weights and biases, a common quantization level in low-power designs that balances accuracy and hardware efficiency without the complexity of more aggressive schemes (e.g., 4 bits). The membrane potential is stored in 9 bits to accommodate the accumulation of multiple weighted inputs, which is particularly important in event-driven operation. A 7-bit counter encodes simulation time (up to 128 steps), while a 3-bit input address field supports the 8 input channels used in our experiments.

A. Analysis Process

The system simulation and performance analysis consist of three stages: Functional Simulation evaluates latency across varying temporal sparsity levels to assess neuron processing time; Post-Synthesis Simulation gathers switching activity data - using Switching Activity Interchange Format (SAIF) - from synthesized architectures for power consumption analysis; Implementation Analysis examines hardware implementation to measure power consumption and resource utilization under different conditions.

B. Analysis Process - Latency

To evaluate the impact of temporal sparsity on computational performance, simulations measured the total processing time while varying the sparsity levels and recording the overall latency for each configuration. In our experiments, architectures that process data serially showed constant latency across the sparsity sweep, because they process every time-step regardless of how many inputs are active. In contrast, the AER neuron was directly influenced by input sparsity, so to capture this effect, simulations were conducted at different levels (25%, 50%, 75%, and 100%). Both decay evaluation methods configurations yielded identical latency performance. Therefore, we focused our analysis on the multiplier-based architecture.

The left-hand side of Figure 3 presents two related views. The top panel shows the absolute latency curves: the clock-driven model consistently exhibits the highest latency, especially under low time sparsity conditions since it processes all time steps without optimization. In contrast, the AER neuron, which selectively processes only active inputs, achieves markedly lower latency when Inputs are more sparse. However, as input density increases, the performance advantage of the event-driven approach diminishes.

The bottom panel unifies the analysis by plotting the ratio $R_{i,k}(\text{Sparsity}) = \frac{L_{\text{event}_i}}{L_{\text{clock}_k}}$ where *i* denotes different event-driven configurations and *k* represents the two clock-driven configurations (multiplier/shifter-based). This formulation explicitly highlights the comparison across multiple configurations for both paradigms. This ratio clearly illustrates the effect of input sparsity: when inputs are mostly inactive, $R_{i,k}(\text{Sparsity})$ is well below 1, highlighting the efficiency of event-driven processing. On the other hand, as input density increases, the ratio approaches or exceeds 1, indicating that the overhead of managing frequent events erodes the latency benefit.

C. Analysis Process - Power

After conducting the performance analysis of the simulated architecture, the design was synthesized, and post-synthesis simulations were carried out to extract the SAIF files. The target platform for implementation was the Pynq-Z2 FPGA, featuring an ARTIX-7 equivalent and a 650 MHz dual-core Cortex-A9 processor. The synthesis effort adhered to Vivado's standard level specifications. Power consumption measurements were performed using the generated SAIF files under stimulus conditions identical to those in the simulation phase. This analysis was conducted on both implementations, as the different hardware solutions impact the power consumption metrics.

Vivado Power Analysis considers two primary contributions to total power consumption: Static and Dynamic Power. However, while Vivado reports a static power of 106 mW across all configurations, this consistency is expected. Static power on the Artix-7 FPGA is determined by device-level characteristics such as leakage current and clocking infrastructure, which remain unchanged given our small design footprint. All neuron variants occupy only a fraction of the fabric, so variations in logic utilization (e.g., between shifter and multiplier designs) do not meaningfully affect static power estimates. As a result, we excluded static power from our analysis to focus on dynamic contributions, which more accurately reflect architectural differences.

Most of the observed results align with intuitive expectations (e.g., higher input activity leads to increased resource usage). Yet, one finding stands out as unpredictable: in the AER-based architecture, higher Input Sparsity is correlated with increased power consumption. Initially, one might assume that fewer input spikes should reduce overall activity—and

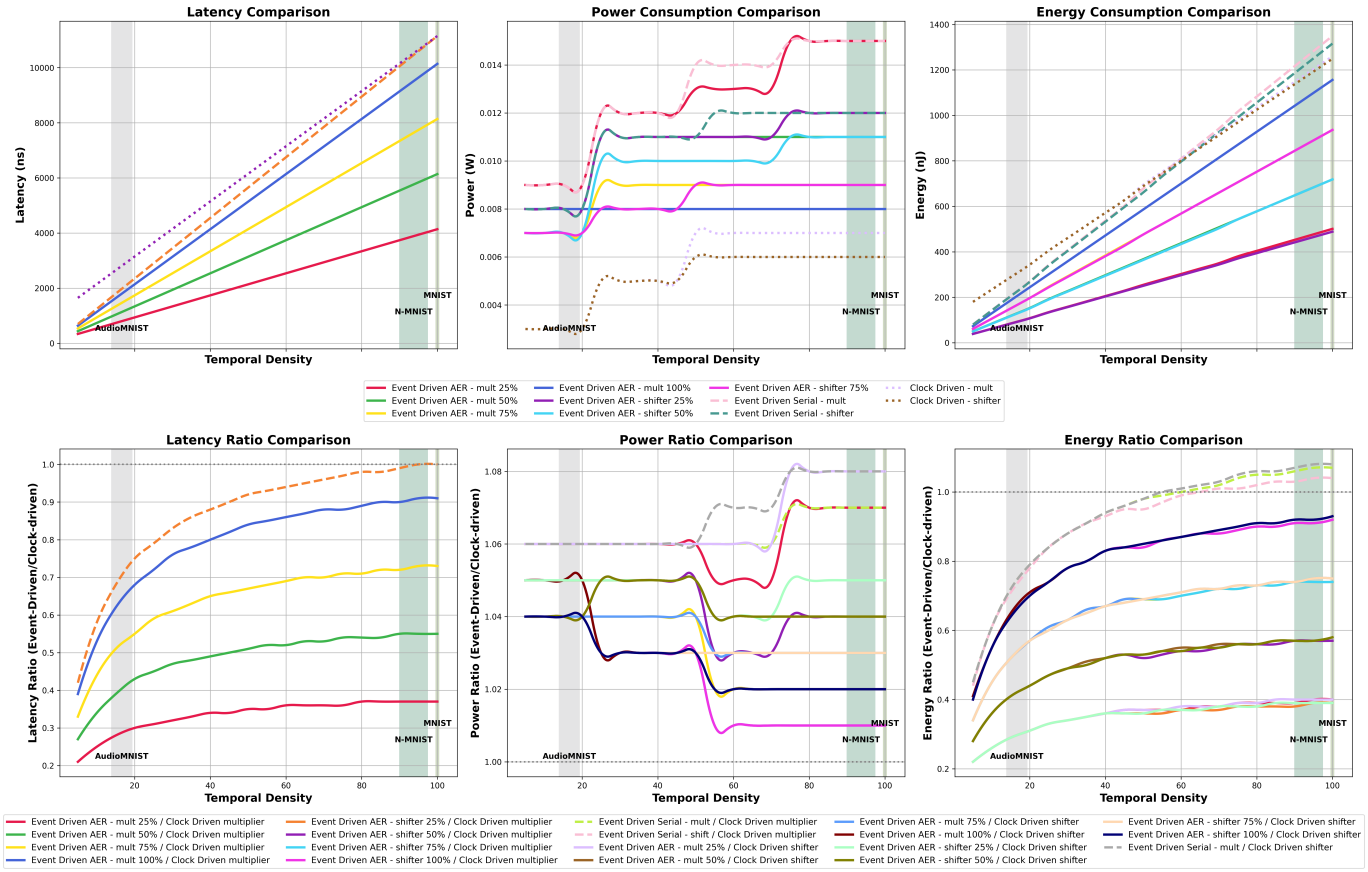

Fig. 3. Comparison of latency, power consumption, and energy across six neuron architectures under varying temporal input density. The top row shows absolute values for all neuron configurations, while the bottom row shows the normalized ratio of event-driven to clock-driven performance (multiplier-based as baseline). Vertical shaded bands mark the average temporal densities of the three evaluated datasets: AudioMNIST (low), N-MNIST (moderate), and MNIST (high). The Figure illustrates the key trade-offs between design choices under realistic sparsity scenarios, emphasizing that energy and latency benefits are most pronounced under sparse input conditions, while power advantages may depend on specific implementation details such as control overhead.

thus lower power—but our measurements show the opposite. A plausible explanation lies in the overhead of the control circuitry. In the serial case, the CU processes a fixed-width input vector at each time step, resulting in predictable control activity that scales linearly with the number of channels. In contrast, in the AER case, the CU remains idle until a packet is detected; when this happens, it triggers a series of actions: decoding the address, initiating memory access, and updating the membrane potential causing a burst of control activity, including FSM transitions and memory interactions. Even though the neuron DP itself processes fewer spikes, the event-based state transitions in the control logic appear to overshadow any power savings from the neuron's reduced arithmetic operations. In other words, the power cost of these "control events" dominates at high input sparsity, producing a counterintuitive increase in total power consumption. Both serial event-driven architectures (multiplier-based and shifter-based) exhibit a trend of rising power consumption as temporal sparsity decreases, which is consistent with a design whose activity scales linearly with the number of input events. Notably, the second column of the first row in Figure 3 clearly demonstrates that the shifter-based design significantly reduces power consumption compared to the multiplier-based approach.

As expected for serial, clock-driven designs, dynamic power rises as input sparsity decreases, even though their latency remains flat (see Section V-B), because more input spikes trigger additional arithmetic activity in each cycle. This increase is due to the more complex logic, including input accumulation and firing conditions, becoming active more frequently as more inputs generate events, contrasting with the simpler decay logic that predominates under high sparsity. While the results reveal no significant difference between the shifter-based and multiplier-based mechanisms for the clock-driven architectures, this is largely due to the precision used by Vivado (0.001W). However, even with this small difference, the power gap becomes more pronounced in a full network implementation, where the impact of power optimization strategies on the entire system is amplified.

D. Analysis Process - Energy Analysis

The last measured metric is energy; it is defined as the product of power and latency ($E = P \times L$). Energy provides a comprehensive quantitative measure of the computational

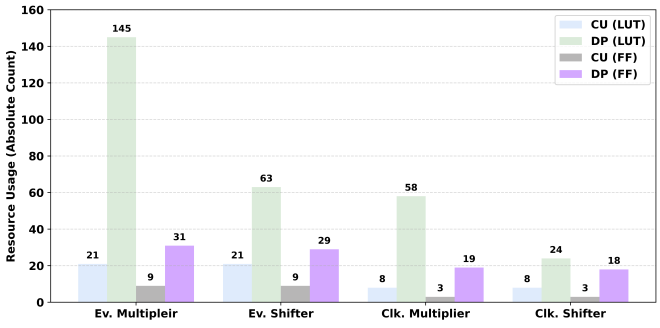

Fig. 4. Resource utilization comparison for different neuron implementations, including event-driven multiplier, event-driven shifter, clock-driven multiplier, and clock-driven shifter. The figure presents the total number of utilized resources, specifically LUTs, and FFs, while also distinguishing their respective contributions to the overall resource consumption.

requirements to process inputs in a neuron beyond a simple comparison of power or latency individually.

Figure 3 (column 3, row 1) shows the absolute energy consumption for each configuration. The AER-based architectures exhibit an increasing trend in energy as the number of active inputs grows, which reflects the heightened activity in the AER control components under denser input conditions. In contrast, the clock-driven and serial architectures follow a similar linear pattern but maintain higher overall energy values under low-sparsity conditions.

E. Resources Utilization analysis

Figure 4 illustrates the distribution of Flip Flops (FFs) and LUTs across the CU and DP for the four designs. In the case of an event-driven neuron, the choice between communication methods (i.e., AER versus serial) does not significantly affect resource usage apart from a difference in the I/O pins of the architecture. For this reason, it was not considered in the comparison. The event-driven architecture using a multiplier exhibits the highest overall resource utilization. In contrast, the event-driven shifter implementation employs significantly fewer LUTs in the DP, while its CU usage remains unchanged compared to the multiplier-based version. Similarly, the clock-driven designs in both configurations show a simpler footprint than the event-driven shifter implementation, and follow the same trend observed between the multiplier- and shifter-based event-driven architectures.

VI. Conclusion and Future Works

This paper compared multiplier-based and shifter-based event-driven LIF neurons against clock-driven implementations, evaluating software and hardware performance across resource usage, latency, power, and accuracy. Although no implementation clearly outperformed the others, results highlight the importance of balancing resource constraints, accuracy, and operating conditions. Future work could integrate these neurons into dedicated hardware accelerators, explore quantization effects, and analyze more complex neuron models to further clarify optimal configurations.

References

[1] A. Carpegna *et al.*, "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. DOI: 10.1109/TETC.2024.3511676.

[2] M. Isik, *A Survey of Spiking Neural Network Accelerator on FPGA*, arXiv:2307.03910 [cs], Jul. 2023. DOI: 10.48550/arXiv.2307.03910. [Online]. Available: http://arxiv.org/abs/2307.03910 (visited on 03/13/2025).

[3] R. Brette *et al.*, "Simulation of networks of spiking neurons: A review of tools and strategies," *Journal of computational neuroscience*, vol. 23, pp. 349–398, 2007.

[4] F. Pavanello *et al.*, "Special Session: Neuromorphic hardware design and reliability from traditional CMOS to emerging technologies," in *2023 IEEE 41st VLSI Test Symposium (VTS)*, arXiv:2305.01818 [cs], Apr. 2023, pp. 1–10. DOI: 10.1109/VTS56346.2023.10139932. [Online]. Available: http://arxiv.org/abs/2305.01818 (visited on 03/03/2025).

[5] K. Yamazaki *et al.*, "Spiking Neural Networks and Their Applications: A Review," en, *Brain Sciences*, vol. 12, no. 7, p. 863, Jul. 2022, Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3425. DOI: 10.3390/brainsci12070863. [Online]. Available: https://www.mdpi.com/2076-3425/12/7/863 (visited on 03/03/2025).

[6] D. Padovano *et al.*, "Spikeexplorer: Hardware-oriented design space exploration for spiking neural networks on fpga," *Electronics*, vol. 13, no. 9, 2024, ISSN: 2079-9292. DOI: 10.3390/electronics13091744. [Online]. Available: https://www.mdpi.com/2079-9292/13/9/1744.

[7] A. L. Hodgkin *et al.*, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, Aug. 1952, ISSN: 0022-3751. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/ (visited on 02/24/2025).

[8] R. Brette *et al.*, "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity," *Journal of Neurophysiology*, vol. 94, no. 5, pp. 3637–3642, Nov. 2005, Publisher: American Physiological Society, ISSN: 0022-3077. DOI: 10.1152/jn.00686.2005. [Online]. Available: https://journals.physiology.org/doi/full/10.1152/jn.00686.2005 (visited on 03/03/2025).

[9] A. Carpegna *et al.*, "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.

[10] P. Lichtsteiner *et al.*, "A 128\times 128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008, Conference Name: IEEE Journal of Solid-State Circuits, ISSN: 1558-173X. DOI: 10.1109/JSSC.2007.914337. [Online]. Available: https://ieeexplore.ieee.org/document/4444573 (visited on 03/03/2025).

[11] S.-C. Liu *et al.*, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISSN: 2158-1525, May 2010, pp. 2027–2030. DOI: 10.1109/ISCAS.2010.5537164. [Online]. Available: https://ieeexplore.ieee.org/document/5537164 (visited on 03/03/2025).

[12] J. K. Eshraghian *et al.*, "Training Spiking Neural Networks Using Lessons From Deep Learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, Sep. 2023, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/JPROC.2023.3308088. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10242251 (visited on 03/03/2025).

[13] A. Paszke *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, arXiv:1912.01703 [cs], Dec. 2019. DOI: 10.48550/arXiv.1912.01703. [Online]. Available: http://arxiv.org/abs/1912.01703 (visited on 03/03/2025).

[14] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Nov. 2012, Conference Name: IEEE Signal Processing Magazine, ISSN: 1558-0792. DOI: 10.1109/MSP.2012.2211477. [Online]. Available: https://ieeexplore.ieee.org/document/6296535 (visited on 03/03/2025).

[15] *Frontiers — Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades*. [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2015.00437/full (visited on 02/26/2025).

[16] S. Becker *et al.*, *AudioMNIST: Exploring Explainable Artificial Intelligence for Audio Analysis on a Simple Benchmark*, arXiv:1807.03418 [cs], Nov. 2023. DOI: 10.48550/arXiv.1807.03418. [Online]. Available: http://arxiv.org/abs/1807.03418 (visited on 02/26/2025).

[17] S. Gupta *et al.*, *FPGA Implementation of Simplified Spiking Neural Network*, arXiv:2010.01200 [cs], Oct. 2020. DOI: 10.48550/arXiv.2010.01200. [Online]. Available: http://arxiv.org/abs/2010.01200 (visited on 03/04/2025).

[18] H. Liu *et al.*, "A Low Power and Low Latency FPGA-Based Spiking Neural Network Accelerator," in *2023 International Joint Conference on Neural Networks (IJCNN)*, ISSN: 2161-4407, Jun. 2023, pp. 1–8. DOI: 10.1109/IJCNN54540.2023.10191153. [Online]. Available: https://ieeexplore.ieee.org/document/10191153 (visited on 03/04/2025).