SCUOLA DI DOTTORATO
UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

Department of Physics "Giuseppe Occhialini"

PhD program in Physics and Astronomy Cycle XVIII

A Reconfigurable TDM Hardware Accelerator for adLIF Spiking Neural Networks

Surname La Gala Name Andrea

Registration number 802726

Tutor: Prof. Mario Zannoni

Supervisor: Prof. Marcello De Matteis

Coordinator: Prof. Laura D'Alfonso

ACADEMIC YEAR 2024/2025

Abstract

Traditional frame-based computer vision paradigms face significant limitations in real-time edge computing due to temporal redundancy and high data bandwidth requirements. Dynamic Vision Sensors (DVSs) offer a solution by mimicking biological retinas to provide asynchronous, event-driven data acquisition with microsecond precision and high dynamic range. However, there is a lack of specialized hardware architectures capable of executing these networks without sacrificing the inherent sparsity and low-power advantages of the spiking paradigm.

This thesis proposes a highly customizable, area-efficient digital SNN accelerator specifically designed to implement adaptive Leaky Integrate-and-Fire (adLIF) neuron dynamics. Unlike standard LIF models, the adLIF model incorporates spike-frequency adaptation (SFA) through a current-based feedback mechanism, providing richer temporal features and intrinsic activity stabilization. The architecture utilizes a Time Division Multiplexing (TDM) execution model, employing a single physical Processing Element (PE) to sequentially update virtual neuron states. This virtualization strategy allows for runtime reconfigurability of arbitrary fully connected topologies, constrained only by internal SRAM capacity. To optimize throughput, the design features a hardware pipeline that overlaps asynchronous event ingestion, memory transfers, and multi-layer processing.

The system was implemented in SystemVerilog and targeted at an AMD Artix-7 FPGA. Experimental results demonstrate a highly efficient hardware footprint, requiring only 5,507 LUTs, 4,668 FFs and 3 DSP slices. The accelerator was validated using the PokerDVS and N-MNIST benchmarks, achieving 91.16% accuracy on N-MNIST using a 12-bit/8-bit fixed-point quantization strategy. Operating at 100 MHz with a total power consumption of 311 mW, the system supports a throughput between 1.27 - 5.08 MEPS for the tested topologies, providing a robust solution for real-time neuromorphic event recognition at the power-constrained edge.

Abstract

I paradigmi tradizionali di visione artificiale basati su frame presentano limitazioni significative nelle applicazioni di edge computing in tempo reale, a causa della ridondanza temporale e degli elevati requisiti di larghezza di banda. I Dynamic Vision Sensors (DVS) offrono una soluzione efficace. Emulando la retina biologica forniscono un'acquisizione dati asincrona e guidata dagli eventi (event-driven), garantendo una precisione al microsecondo e un'elevata gamma dinamica. Tuttavia, si riscontra una carenza di architetture hardware specializzate, capaci di processare i dati provenienti da questi sensori, senza sacrificare la sparsità intrinseca e i vantaggi di basso consumo tipici del paradigma spiking.

Questa tesi propone un acceleratore digitale per Spiking Neural Networks (SNN), altamente personalizzabile ed efficiente in termini di occupazione d'area, progettato specificamente per implementare la dinamica neuronale di tipo adaptive Leaky Integrate-and-Fire (adLIF). A differenza dei modelli LIF standard, il modello adLIF integrando la Spike Frequency Adaptation (SFA) attraverso una corrente di feedback, fornendo caratteristiche temporali più ricche e una stabilizzazione intrinseca durante la fase di addestramento.

L'architettura sfrutta un modello di esecuzione basato sul multiplexing a divisione di tempo (TDM), impiegando una singola unità di elaborazione fisica (PE) per aggiornare sequenzialmente gli stati dei neuroni virtuali. Questa strategia di virtualizzazione consente la riconfigurabilità a runtime di topologie fully connected arbitrarie, con il solo vincolo della capacità della SRAM interna. Per ottimizzare il throughput, il design include una pipeline hardware che sovrappone la ricezione asincrona degli eventi, i trasferimenti di memoria e l'elaborazione dei layer.

Il sistema è stato implementato in SystemVerilog e sintetizzato su FPGA AMD Artix-7. I risultati sperimentali dimostrano un'elevata efficienza nell'uso delle risorse, richiedendo solo 5.507 LUT, 4.668 FF e 3 DSP. L'acceleratore è stato validato tramite i benchmark PokerDVS e N-MNIST, raggiungendo un'accuratezza del 91,16% su N-MNIST con una strategia di quantizzazione a virgola fissa a 12/8 bit. Operando a 100 MHz con un consumo di potenza

totale di 311 mW, il sistema supporta un throughput compreso tra 1,27 e 5,08 MEPS per le topologie testate, offrendo una soluzione robusta per il riconoscimento di eventi neuromorfici in tempo reale in contesti edge con vincoli energetici stringenti.

Contents

List of Acronyms

adLIF Adaptive Leaky Integrate and Fire

AER Address Event Representation

AHB Advanced High-performance Bus

ASIC Application-Specific Integrated Circuit

CNN Convolutional Neural Network

DVS Dynamic Vision Sensor

EoF End of Frame

EPS Events Per Second

FIFO First In First Out

FPGA Field Programmable Gate Array

FSM Finite State Machine

GPIO General-Purpose Input/Output

LIF Leaky Integrate and Fire

MMCM Mixed-Mode Clock Manager

MSB Most Significant Bit

PE Processing Element

PTQ Post Training Quantization

QAT Quantization Aware Training

RTL Register Transfer Level

SFA Spike Frequency Adaptation

SFM Spike Frequency Modulation

SIMD Single Instruction Multiple Data

SNN Spiking Neural Network

SNR Signal-to-Noise Ratio

SoC System-on-Chip

SRAM Static Random-Access Memory

STDP Spike-timing-dependent plasticity

TDM Time Division Multiplexing

Chapter 1

Introduction

This chapter illustrates the motivations for transitioning from traditional frame-based computer vision to event-driven neuromorphic architectures. It begins with an analysis of the systemic inefficiencies of conventional CMOS sensors, specifically regarding temporal redundancy and high-speed data bottlenecks.

To address these limitations, Dynamic Vision Sensors (DVSs) are introduced as a biologically-inspired alternative capable of achieving microseconds temporal resolution and superior dynamic range. The discussion then shifts to the processing gap, justifying the adoption of Spiking Neural Networks (SNNs) over traditional Convolutional Neural Networks (CNNs) to maintain the inherent sparsity and timing information of the DVS stream.

The core of this work is based on the implementation of an SNN utilizing the Adaptive Leaky Integrate and Fire (adLIF) neuron model. Consequently, a mathematical derivation of its continuous-time dynamics is provided, along with its subsequent discretization via the Semi-Implicit Euler method. This mathematical framework forms the basis for the hardware-oriented architecture described in the following chapters.

Finally, the thesis objectives and structural organization are outlined.

1.1 Dynamic Vision Sensors

The proliferation of real-time computer vision and edge computing has exposed the fundamental limitations of traditional frame-based paradigms [1, 2, 3]. Standard CMOS image sensors operate by integrating light over a fixed exposure time across the entire pixel array, typically at rates of 30 to 60 frames per second (fps) [1, 4]. This synchronous approach introduces two primary systemic inefficiencies:

Table 1.1: Quantitative Comparison between Conventional CMOS and DVS. Data highlights the architectural advantages of event-based acquisition in latency, dynamic range, and power efficiency [10, 12].

Metric	Conventional CMOS	Dynamic Vision Sensor
Temporal Resolution	∼16.6 ms (at 60 fps)	∼1 μs
Latency	High (>10 ms)	Low (<100 μs)
Dynamic Range	60 – 80 dB	120 – 140 dB
Power Consumption	100 mW – 1 W	10 – 100 mW
Data Volume	Constant / High	Sparse / Activity-Dependent

- **Temporal Redundancy**: In typical scenes, large spatial regions remain static between frames, such as the background [5]. Frame-based sensors indiscriminately sample these areas, generating a massive volume of redundant data that saturates the processing pipeline with static information, which must be discarded by subsequent algorithms [6, 1].
- **The High-Speed Trade-off**: Capturing fast-moving objects necessitates minimized exposure times to avoid motion blur [7]. However, increasing the frame rate to improve temporal resolution results in an increase in data bandwidth requirements, much of which remains non-informative [8]. Furthermore, reduced integration times significantly degrade the Signal-to-Noise Ratio (SNR) and sensitivity [9].

DVSs represent a significant shift toward efficient, **event-driven data acquisition**. By mimicking the biological retina, DVSs employ independent, **asynchronous pixels** that respond only to local changes in brightness at the single-pixel level [10, 11]. This mechanism filters out redundant information at the source, transmitting data only when a change in log-intensity exceeds a preset threshold [10].

As illustrated in Figure 1.1, the sensor output can be visualized by accumulating events over time: pixels with decreasing brightness (negative polarity) appear darker, while those with increasing brightness (positive polarity) appear lighter. As summarized in Table 1.1, this shift in acquisition results in superior performance across major metrics:

- **Temporal Resolution and Latency**: Since pixels operate independently, the DVS captures rapid dynamics with **microsecond precision**, avoiding the motion blur typical of frame-based cameras [12, 11].

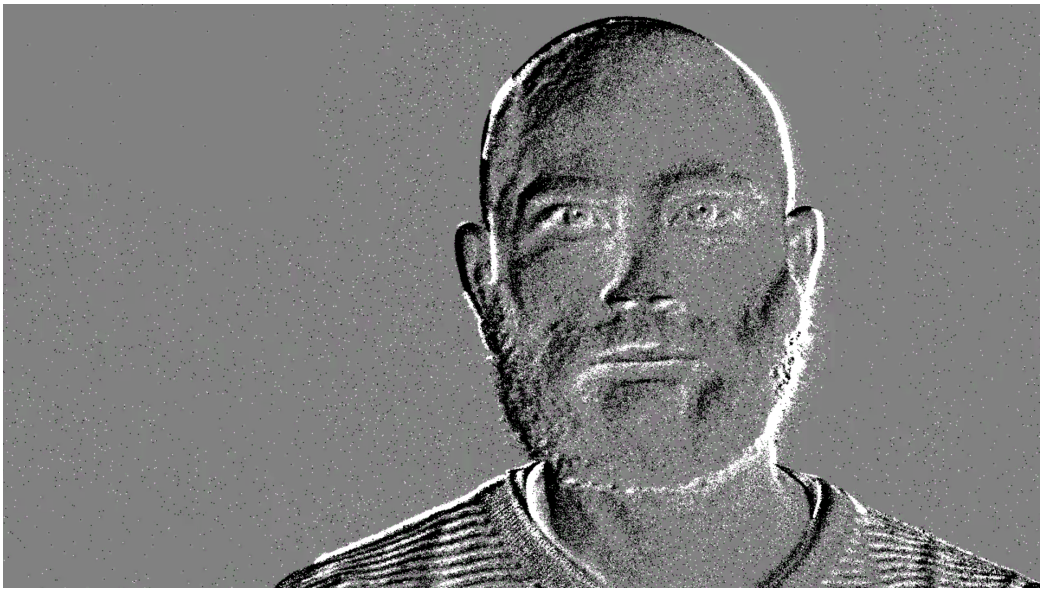

Figure 1.1: Visualization of DVS Event Data. The image represents the spatial accumulation of asynchronous events over a time window. White pixels indicate a positive change in brightness (ON events), dark pixels indicate a decrease (OFF events). The grey background represents regions without activity. Unlike traditional frames, these areas remain empty because the sensor only responds to dynamic temporal changes.

- **Dynamic Range**: The logarithmic response of the pixels enables operation in high-contrast environments (HDR up to 120-140 dB) where standard sensors fail due to saturation or lack of sensitivity [10].
- **Data Sparsity**: Transmitting only relevant changes reduces the bitrate and improves the efficiency of the entire processing chain, as the processor is no longer required to discard useless data manually [11].

The primary output of a DVS is a stream of discrete, **asynchronous events** rather than a sequential array of intensity values. Each event e_k is defined as a four-tuple $\{x_k, y_k, t_k, p_k\}$, where x, y are the spatial coordinates, t is the microsecond-resolution timestamp, and $p \in \{-1, +1\}$ is the polarity indicating a decrease or increase in brightness.

The use of Address Event Representation (AER) [13] allows the sensor to communicate these events asynchronously and exploit **data sparsity**. This sparsity is the primary driver for the efficiency of SNN architectures [2], as it enables a direct mapping between sensor activity and hardware state updates.

As will be demonstrated in Section 4.4.1, the processing latency of the

proposed Time Division Multiplexing (TDM) based accelerator scales linearly with this input activity, effectively exploiting the temporal sparsity inherent in neuromorphic data to minimize energy consumption.

1.2 The Neuromorphic Approach

Because the DVS output is composed of a sequence of asynchronous events, rather than traditional intensity images, conventional computer vision algorithms cannot be directly applied [14, 15]. Standard Machine Learning models, such as CNNs, require a structured tensor inputs, often forcing the accumulation of events into dense "**pseudo-frames**"[14]. This conversion introduces a fundamental inefficiency: to process a pseudo-frame, a conventional processor must perform redundant computations across the entire spatial grid, including regions where no events occurred.

SNNs provide a more coherent alternative by aligning the processing paradigm with the sparse nature of the input. This architecture enables **event-driven computation**, which drastically alters the power profile of the system. In a CNN, the computational cost is constant and determined by the layer dimensions, regardless of the scene content. In contrast, in an SNN, neurons remain idle and consume low dynamic power unless they receive an input spike [15]. In scenarios where the scene is stationary the theoretical computational cost and data movement drop to near zero [14], providing a direct path to energy efficiency at the edge.

Furthermore, **SNNs incorporate time into the network state** through the internal evolution of the neuronal membrane potential. Unlike CNNs, which are inherently memoryless between discrete inferences, the adLIF neurons in this work act as a dynamical system where information is not only encoded in the spatial distribution of spikes but also in their relative timing. Consequently, SNNs can exploit **temporal correlations** naturally [16]. A neuron can integrate multiple sparse events occurring at different timestamps to reach its firing threshold, effectively performing temporal feature extraction without the need for frame-based buffering.

The integration of event-based sensors with spiking architectures enables a streamlined **neuromorphic processing chain**. This pipeline maintains a functional flow where processing remains driven by the timing of input events. While digital implementations require temporal discretization to manage neuronal dynamics, such as membrane potential decay and synaptic integration, this can be performed at a resolution that preserves the salient temporal features of the DVS signal [17].

Despite the theoretical advantages of SNNs, a significant gap remains

between high-level algorithmic simulations and efficient hardware implementations [14]. Many existing hardware-optimized models prioritize area efficiency over the sophisticated neuronal dynamics that drive high-level recognition tasks [18]. Consequently, there is a lack of specialized hardware capable of supporting more advanced models, such as the adaptive Leaky Integrate-and-Fire (adLIF) neuron. The objective of this work is to bridge this gap by proposing a hardware architecture that accommodates these richer dynamics. By implementing the adLIF model, it aims to improve classification accuracy [19], ensuring that the benefits of event-driven sensing are preserved without being limited by oversimplified neuronal logic.

1.3 The Hardware-Oriented adLIF Model

To implement SNNs in digital hardware, biological models must be translated into computationally efficient equations. While the Leaky Integrate and Fire (LIF) model is a common hardware baseline due to its minimal resource requirements, it fails to capture critical temporal features such as Spike Frequency Adaptation (SFA) [19, 20]. To overcome this limitation, this work adopts the adLIF model.

Adaptation in SNNs can be implemented either via a **dynamic threshold** [21] or a **feedback current** [22, 19]. In this work, a current-based mechanism was selected as it demonstrated superior classification accuracy in literature [22, 19]. Figure 1.2 compares the dynamics of LIF and adLIF models, showing how the addition of an adaptation variable allows the neuron to modulate its firing rate in response to persistent stimuli, providing a more biologically plausible and computationally powerful processing unit compared to the standard LIF model.

1.3.1 Continuous-Time Dynamics

The adLIF model describes the neuron's state through two coupled first-order differential equations, which represent the evolution of the membrane potential and the internal adaptation state. The first equation (Eq. 1.1) defines the membrane potential $u(t)$ as a leaky integrator of both the synaptic input current $I(t)$ and the internal inhibitory feedback current $w(t)$:

$$\tau_u \frac{du(t)}{dt} = -u(t) + I(t) - w(t) \tag{1.1}$$

In this expression, the term $-u(t)$ represents the **passive leakage** of the membrane, which pulls the potential back to its resting state in the absence

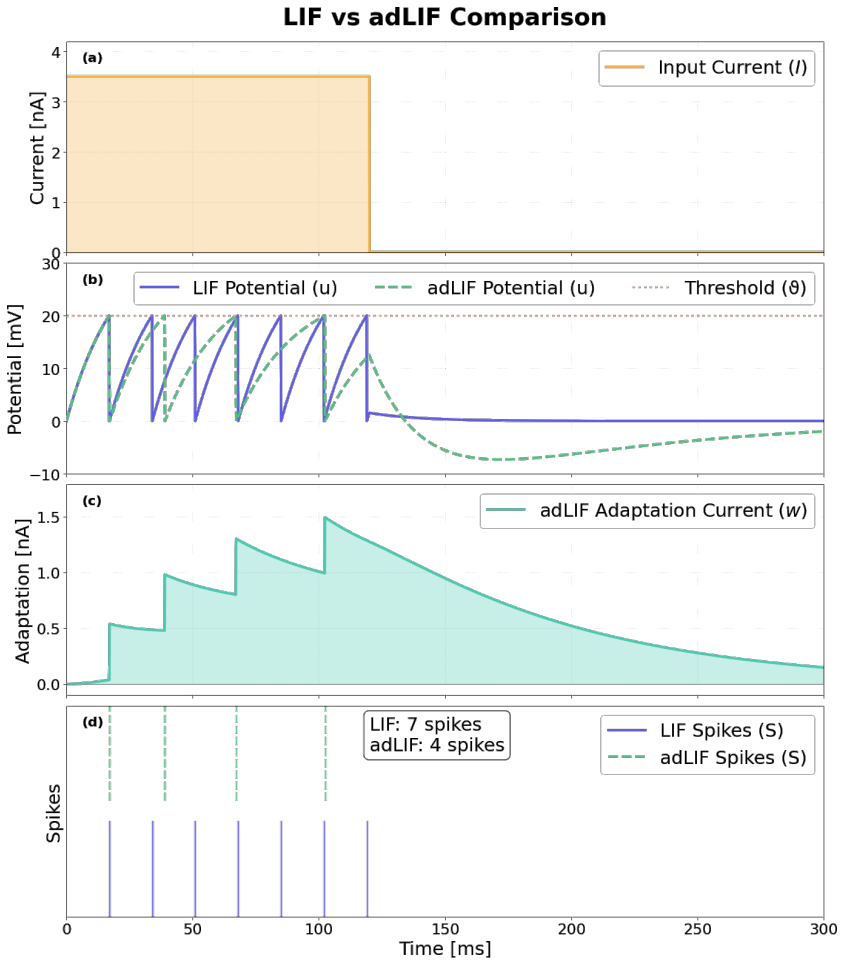

Figure 1.2: Comparison of LIF and adLIF neuron dynamics under constant input stimulus. **(a)** Applied constant input current $I(t)$. **(b)** Evolution of membrane potential $u(t)$ for both models. **(c)** Adaptation current $w(t)$ providing negative feedback, which increases with each spike and provides negative feedback. **(d)** Output spiking activity $S(t)$, illustrating the spike-frequency adaptation (SFA) effect in the adLIF neuron compared to the regular firing pattern of the LIF model.

of input. The inclusion of $-w(t)$ is the defining feature of the adaptive model: it acts as a **subtractive feedback** that effectively raises the effective firing threshold over time or suppresses the integration rate based on previous activity.

The second equation (Eq. 1.2) tracks the adaptation variable $w(t)$, which evolves according to:

$$\tau_w \frac{dw(t)}{dt} = -w(t) + a \cdot u(t) + b \cdot \tau_w \sum_{t_s} \delta(t - t_s) \tag{1.2}$$

This adaptation current is driven by two distinct factors. The term $a \cdot u(t)$ represents **sub-threshold adaptation**, where the adaptation current grows proportionally to the membrane potential even before a spike is generated. This allows the neuron to exhibit resonance and frequency-selective filtering. The second term, $b \cdot \tau_w \sum_{t_s} \delta(t - t_s)$, represents spike-triggered adaptation. Each time the neuron fires (at times t_s), the adaptation variable $w(t)$ is increased by a discrete amount b.

Mathematically, τ_u and τ_w are the time constants for the potential and adaptation current respectively, where usually $\tau_w \gg \tau_u$ to allow the adaptation to track the history of the neuron over a longer temporal window. This separation of time scales is what enables the model to exhibit SFA, gradually reducing the firing frequency in response to a constant stimulus. This feedback loop modulates the spiking activity based on the neuron's historical activation, providing a form of **gain control** that is intrinsic to the neuron's own dynamics.

The coupling of these dynamics allows the adLIF model to exhibit a significantly richer set of temporal behaviors compared to the standard LIF model. Specifically, the interaction between the membrane potential and the sub-threshold adaptation enables the neuron to sustain **sub-threshold membrane potential oscillations**. These oscillations transform the neuron's functional role from a simple integrator into a **resonator** [19].

This resonance makes the adLIF neuron inherently **frequency-selective**, allowing it to respond most strongly when the periodicity of input spikes aligns with its intrinsic oscillation frequency [23]. From a signal processing perspective, this property is particularly advantageous for detecting Spike Frequency Modulation (SFM) [24], a common feature where information is embedded in the temporal evolution of firing rates rather than just instantaneous events.

Beyond feature extraction, the adaptation mechanism serves a critical role in network training and hardware efficiency. While high-performance SNNs often rely on external normalization techniques, such as **Batch Normalization**, to prevent exploding gradients and maintain stability [25, 26], the negative feedback loop inherent in the adLIF dynamics provides **intrinsic activity stabilization** [19]. By self-regulating the firing rate through the $w(t)$ variable, the network achieves robust performance without requiring the additional computational overhead and memory buffers associated with explicit normalization layers in the hardware datapath. This makes the adLIF model not only more computationally capable but also highly compatible with

the resource-constrained nature of Field Programmable Gate Array (FPGA) implementations.

1.3.2 Discrete-Time Derivation via Semi-Implicit Euler

Digital systems operate in discrete time-steps Δt. To discretize Eq. 1.1 and 1.2, the **Semi-Implicit Euler method**[27] is employed. Unlike the standard Forward Euler, the semi-implicit approach updates the second state variable ($w[t]$) using the already-updated value of the first ($u[t]$), offering better **numerical stability** for coupled systems. This method captures the feedback loop dynamics without requiring extremely small time-steps [19].

By defining the decay factors as $\alpha = 1 - \frac{\Delta t}{\tau_u}$ and $\beta = 1 - \frac{\Delta t}{\tau_w}$, the discrete-time evolution is formulated as:

$$\begin{cases} u[t] = \alpha u[t-1] + (1-\alpha)(I[t] - w[t-1]) \\ w[t] = \beta w[t-1] + (1-\beta)(a \cdot \hat{u}[t] + b \cdot S[t]) \end{cases} \tag{1.3}$$

where $I[t]$ is assumed constant over the interval Δt, and $\hat{u}[t]$ represents the membrane potential after the reset mechanism is applied. In a multi-layer digital architecture, the input current $I[t]$ for a specific neuron j is the accumulation of weights from the set of active presynaptic neurons i, expressed as:

$$I_j[t] = \sum_i S_i[t] \cdot W_{i,j} \tag{1.4}$$

Where $S_i[t]$ is 1 if neuron i spiked and $W_{i,j}$ is the weight associated with the connection from neuron i to neuron j. In this work, the spiking process is followed by a **reset-by-subtraction** (or "**soft reset**") mechanism. Once the membrane potential $u[t]$ exceeds the threshold ϑ, an output spike $S[t] = 1$ is registered, and the potential is immediately updated for the next cycle. Instead of clearing the state, the threshold value is subtracted from the current potential:

$$\hat{u}[t] = \begin{cases} u[t] - \vartheta & \text{if } u[t] \geq \vartheta \quad (S[t] = 1) \\ u[t] & \text{if } u[t] < \vartheta \quad (S[t] = 0) \end{cases} \tag{1.5}$$

Compared to the "**reset-to-zero**" (**hard reset**) approach, the soft reset is **less lossy**: it preserves the residual information ($u[t] - \vartheta$), which has been shown to improve classification accuracy [19].

1.4 Thesis Contributions and Organisation

This work addresses the gap between algorithmic SNN simulations and efficient hardware by proposing a SNN architecture for high-speed dynamic event recognition. The main contributions of this thesis are:

- **Design and Validation of a Digital SNN Accelerator**: An architecture based on TDM is proposed. It utilizes a single physical Processing Element (PE) to manage any fully connected topology by sequentially updating virtual neuron states.
- **Hardware Implementation of the adLIF Model**: To the best of our knowledge, this work represents one of the first FPGA-based accelerators to specifically implement the **adaptive Leaky Integrate-and-Fire** dynamics. By incorporating SFA, the architecture provides the network with a richer set of temporal features than standard LIF designs.
- **Evaluation on Neuromorphic datasets**: The architecture is validated using the **N-MNIST** and **PokerDVS** benchmarks. Results demonstrate the system's ability to classify event based databases with an **AMD Artix-7 FPGA**, achieving competitive accuracy while maintaining a low hardware footprint.

The remainder of this thesis is organized as follows:

- **Chapter 2** presents the hardware architecture of the adLIF accelerator. It details the TDM execution model, the memory organization aimed at optimizing bandwidth, and the design of the PE tailored for the discretized adLIF dynamics.
- **Chapter 3** discusses the training methodology and the quantization strategy. It introduces the PokerDVS and N-MNIST neuromorphic datasets, describes the statistical analysis used for temporal window selection, and validates the 12-bit fixed-point representation through a bit-accurate golden model to evaluate the accuracy drop.
- **Chapter 4** reports the experimental results of the implementation on the AMD Artix-7 FPGA. It analyzes resource utilization, power consumption breakdown, and classification accuracy, concluding with a comparison against other FPGA-based SNN accelerators.
- **Chapter 5** summarizes the main contributions of this work and outlines potential future research directions.

Chapter 2

Hardware Architecture of the adLIF Accelerator

This chapter provides the technical description of the Adaptive Leaky Integrate and Fire (adLIF) accelerator. The discussion begins with the high-level system integration, detailing the integration of the internal memory hierarchy, the Advanced High-performance Bus (AHB) infrastructure [28], and the core Processing Element (PE). Central to the architecture is a Time Division Multiplexing (TDM) execution model, which utilizes a single physical PE to sequentially update virtual neuron states. This virtualization strategy enables support for arbitrary fully connected topologies constrained only by the available on-chip memory.

The system's communication interfaces are examined, highlighting the dual-path support for Address Event Representation (AER) [13] for real-time sensor integration and an AHB-Lite bus for configuration and diagnostic monitoring. Subsequent sections detail the memory organization and mapping strategies used to store network parameters and synaptic weights. The focus then shifts to the internal architecture of the PE, responsible for the system-state updates. The chapter concludes with a review of the architectural trade-offs, evaluating the balance between computational throughput, memory footprint, and hardware complexity.

2.1 System Architecture

The high-level architecture of the proposed accelerator is illustrated in Figure 2.1. At the core of the design is the **TDM execution model** [29], which employs a single physical PE to sequentially update the states of all **virtual neurons** within the network . This virtualization strategy allows the hardware

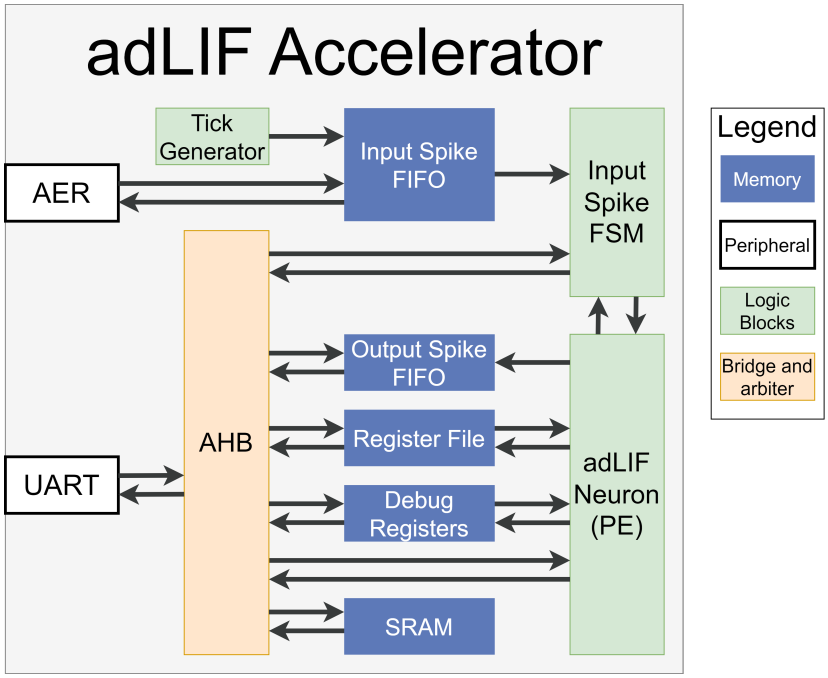

Figure 2.1: Top-level block diagram of the adLIF Spiking Neural Network (SNN) Accelerator. The system utilizes an AHB-Lite infrastructure for configuration and external access, while the PE maintains dedicated low-latency paths for neural processing.

to support any **fully connected** topology based on the adLIF model, constrained only by the available capacity of the internal Static Random-Access Memory (SRAM).

The system is organized into three primary functional domains:

- **Event Acquisition and Synchronization**: Composed of the Tick Generator, the Input Spike FIFO, and the Input Spike Finite State Machine (FSM). This domain manages the ingestion of asynchronous events and their alignment with the system's global temporal windows.
- **Memory and Bus Infrastructure**: Centered around an AHB-Lite interconnect with an internal arbiter. This subsystem manages concurrent memory requests from the PE, the Input Spike FSM, and external host interfaces, ensuring data integrity across the SRAM and the Register File.
- **Processing Core**: The adLIF PE, which implements the discretized neural dynamics and manages the network update flow.

The internal memory blocks include the primary **SRAM** for weights and parameters, an **Output Spike FIFO** for accumulating classification results, and a **Register File**. The latter stores the first two layer configurations as a fast-access cache for the starting layer and real-time status indicators (e.g., FIFO occupancy, execution states). Additionally, a dedicated set of **Debug Registers** provides direct visibility into internal datapath signals, facilitating cycle-accurate verification and diagnostic monitoring. The full memory map with the registers is presented in Appendix A.1.

By offloading the network description (topology, synaptic weights, and neuronal parameters) to the internal memory rather than hard-coding them at the Register Transfer Level (RTL), the architecture achieves a high degree of **run-time reconfigurability**. This enables the deployment of diverse Spiking Neural Network (SNN) topologies **without the need for hardware re-synthesis**. Furthermore, as illustrated in the block diagram in Figure 2.1, the high-bandwidth paths between the PE, the **Register File**, and the **Spike FIFOs** are implemented as direct interconnects. This dual-path approach effectively decouples the high-speed neural computation from the arbitration overhead of the shared AHB, ensuring that the PE maintains maximum throughput during active inference cycles.

2.1.1 Interfaces and Event Acquisition

The architecture supports two primary communication pathways: a high-speed **AER** interface for real-time sensor integration and an **AHB-Lite bus** for system configuration and host-based testing.

Address Event Representation (AER)

External spikes are captured using the AER protocol, which serves as the standard for asynchronous neuromorphic communication. This protocol allows the accelerator to directly interface with the **asynchronous domain of Dynamic Vision Sensors (DVSs)**. As illustrated in Figure 2.2, the protocol utilizes a **four-phase handshake** mechanism [13]. In this protocol, the sender raises the `req` signal and waits for the receiver `ack`. The event address is sampled by the receiver logic only when both the `req` and `ack` signals are asserted. Upon reception, events are buffered within an internal Input Spike FIFO. This buffer serves as the synchronization boundary, transitioning the data from the asynchronous handshake domain to the system clock domain. Once synchronized, the events are transferred to the internal SRAM via the Input Spike FSM for subsequent processing.

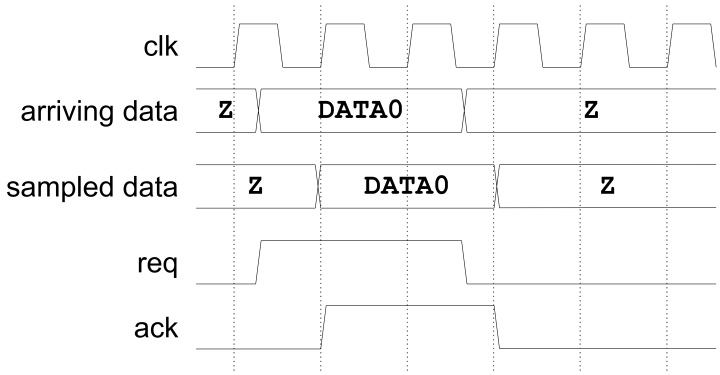

Figure 2.2: Timing diagram of the AER handshake protocol. The four-phase handshake (Request/Acknowledge) allows asynchronous data transfer. The event address (DATA0) is sampled by the receiver logic when both `req` and `ack` signals are asserted.

AHB-Lite and UART-AHB Bridge

For control and configuration, the accelerator acts as a slave on the AHB-Lite bus. In this prototype implementation, communication is managed through a **UART-AHB bridge**, which enables a host PC to interact with the FPGA development board. This secondary path is used for:

- **Spike Injection**: During validation, pre-recorded datasets (such as N-MNIST or PokerDVS) can be injected directly into the SRAM, bypassing the physical AER pins.
- **Weight and Parameter Loading**: Offline-trained synaptic weights and neuronal parameters ($\alpha, \beta, \vartheta, etc.$) are loaded into the SRAM prior to execution.
- **Result Acquisition**: Classification results are read from the Output Spike FIFO once the inference is complete.
- **Real-time monitoring**: The host PC maintains real-time access to status and debug registers, allowing for monitoring of the hardware state.
- **Control**: Control and Status signals stored in Register File are also accessible through the AHB interface

While the current bridge uses UART [30] for development convenience, the modular design ensures that in a System-on-Chip (SoC) environment, the accelerator can be tied **directly to a high-performance system bus** for

configuration while the AER interface manages the low-latency spiking data flow independently.

2.1.2 Spikes Frame Packetization

Events acquired through the AER protocol must be organized into distinct temporal windows to be processed by the hardware. The accelerator performs all state updates and synaptic integrations within these discrete time-steps, which must remain consistent with the Δt used during the training phase [31] to ensure the validity of the learnable decay constants (α, β). The synchronization flow is governed by a **programmable Tick Generator**, which serves as the **temporal heartbeat** of the system. The packetization process operates as follows:

- **Real-time Ingestion**: Asynchronous events are pushed into the Input Spike FIFO immediately upon acquisition via the AER interface.
- **Temporal Delimitation**: The Tick Generator produces a periodic signal at a frequency configured via the Register File, defining the boundary of each processing step.
- **Packet Closure**: Upon the occurrence of a tick, a special **End of Frame (EoF) marker** is inserted into the FIFO, effectively "closing" the current temporal frame.

The structure of the data stored in the Input Spike FIFO is illustrated in Figure 2.3. As shown in the diagram, each frame consists of the sequence of event addresses accumulated during the window. The **EoF marker** is identified by its **Most Significant Bit (MSB)** being set to '1'. The remaining bits of the marker contain the **total spike count** for that specific frame. This value acts as a check variable, allowing the Input Spike FSM to verify that no events were lost during the transfer from the acquisition interface to the SRAM.

Data Transfer and Memory Buffering

The movement of events from the input FIFO to the SNN core is managed by an **autonomous Input Spike FSM**. This unit acts as an **AHB master**, responsible for transferring the buffered batches of events to a dedicated region in the system SRAM. This buffering stage decouples the variable arrival rate of external asynchronous data from the high-speed synchronous execution of the PE.

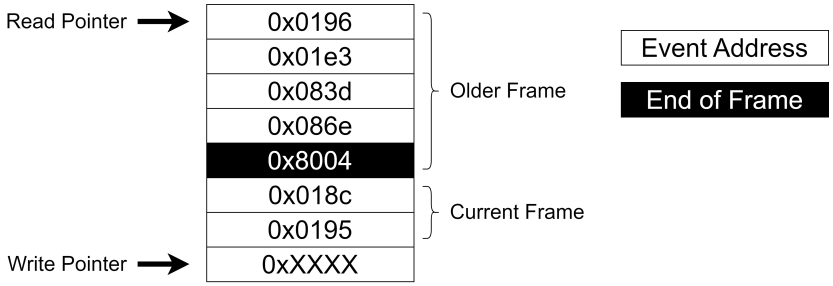

Figure 2.3: Internal Structure of the Input Spike FIFO. Spikes are stored as a sequence of event addresses. Each temporal frame is terminated by an End of Frame (EoF) marker, identified by a Most Significant Bit (MSB) of 1, which also contains the total spike count for that window to facilitate error detection.

For AER data, the FSM initiates a transfer loop as soon as the FIFO contains valid data, mapping event addresses directly to the SRAM. To optimize throughput, the FSM can begin transferring a new frame from the FIFO while the PE is still active, provided the PE has completed the update of the current input layer. This significantly reduces the perceived latency of the memory writing process. However, the system architecture **grants the PE a higher level of privilege**: if the PE requires full memory bandwidth for state updates, the FSM stalls until the memory bus is released.

As illustrated in Figure 2.4, a **Transfer Stall** may occur if a frame is not completely transferred before the start of the next processing cycle. This typically occurs when a burst of events arrives immediately before the global tick. During these periods, the PE remains inactive, allowing the Input Spike FSM to utilize burst write requests to clear the remaining FIFO data rapidly. Once the final event of a packet (delimited by the EoF) is stored, the FSM asserts the start signal for the PE, ensuring total data integrity before computation begins.

While the AER path utilizes this FIFO-based packetization, data injected via the AHB-Lite bus (e.g., during N-MNIST validation), is written directly into the SRAM by the host. In this scenario, packet boundaries are implicitly defined by the host's memory operations, bypassing the Input Spike FIFO and the EoF logic. The host must autonomously trigger execution by writing to the appropriate control register in the Register File once the SRAM write is complete.

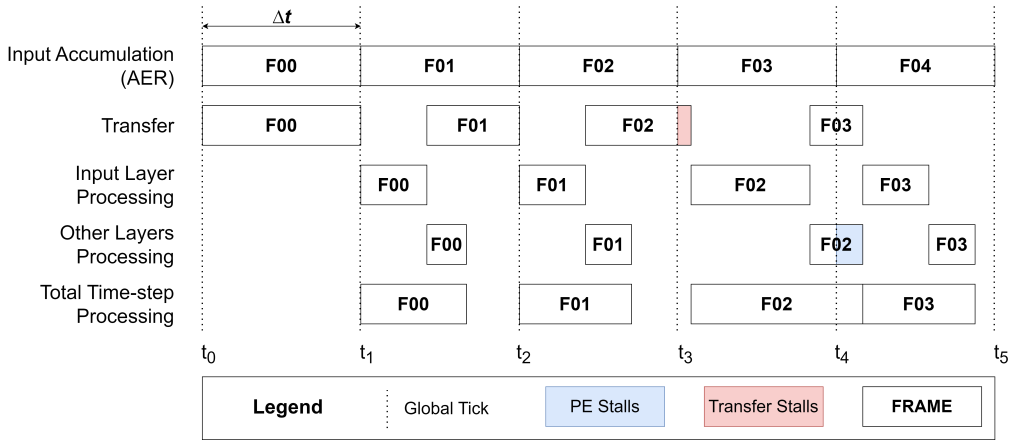

Figure 2.4: Hardware Pipeline Timing Diagram. Illustrates the temporal overlap between Address Event Representation (AER) input accumulation, data transfer to SRAM, and the multi-layer PE execution. The diagram highlights the system's ability to handle variable processing latencies in transfering and processing frames.

Timing Robustness and Backpressure

The FIFO-buffered architecture provides intrinsic **resilience to variable workloads**. Because the time required to process a single time-step is proportional to input activity (the number of active spikes), high-activity bursts may occasionally cause the PE execution time to exceed the nominal global tick Δt, as illustrated near t_4 in Figure 2.4. In these instances, the Input Spike FSM postpones the trigger for the subsequent processing cycle until the current frame's updates are finalized, while incoming events continue to accumulate in the Input Spike FIFO under the correct temporal frame.

This structure implements a natural **backpressure mechanism**: incoming events are buffered rather than dropped during transient peaks. As long as the PE's average processing throughput exceeds the sensor's average event rate, the system utilizes periods of low activity, inherent in the spatial and temporal sparsity of DVS data [32], to clear the buffer. This ensures that the long-term **temporal accuracy** of the SNN is preserved and no event data is lost, even if individual time-steps experience processing jitter.

2.1.3 Memory Organization

The memory map is designed to optimize data bandwidth and minimize inference cycle latency. To ensure adaptability across various platforms (FPGA, ASIC, or simulation environments), a parameterized design is em-

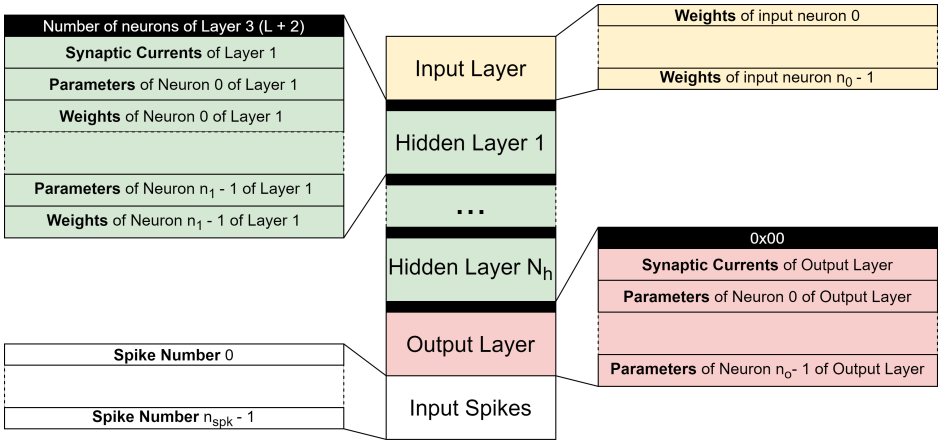

Figure 2.5: SRAM Memory Map and Data Organization. The memory space is organized into contiguous blocks for each network layer to facilitate sequential TDM updates. Each layer block contains specific regions for synaptic weights, neuron parameters, and state variables. At the start of every Layer L, a marker specifies the number of neurons in Layer $L + 2$.

ployed where bit-widths and packing factors are defined via pre-compilation macro directives.

The configuration tested in the following chapters uses a centralized SRAM with **48-bit wide rows**. To maximize throughput, multiple data elements are packed into a single memory word, enabling the system to fetch multiple operands in a single clock cycle. As illustrated in Figure 2.5, the information stored in the SRAM is organized into the following functional regions:

- **Synaptic Weights (w_{ij})**: These are stored in contiguous blocks following the neuron parameters to facilitate burst reading during spike propagation. Each memory line contains multiple weights, with the specific number defined by the W_{pl} packing factor.

- **Synaptic Currents (I)**: These values are packed at the beginning of each layer block. They are organized with I_{pl} synaptic currents per line to enable parallel updates resembling a Single Instruction Multiple Data (SIMD) approach. This ensures that for every memory read operation during a postsynaptic update, multiple weights and currents are retrieved simultaneously.

- **Neuron Parameters and States**: State variables (u, w) and neuronal parameters $(\alpha, \beta, \vartheta, a, b)$ are fetched dynamically. While these elements are currently assigned to full memory lines to facilitate the serialized

TDM update sweep, future iterations will implement packing to further optimize memory density.

- **Layer Information**: To minimize pipeline stalls, the hardware maintains a look-ahead buffer containing the address and neuron count for three consecutive layers: the current presynaptic layer, the active postsynaptic layer, and the upcoming layer. For example, while updating Layer L, the system retrieves a marker specifying the neuron count for Layer $L + 2$ during available free clock cycles. This ensures that at the start of any layer update, the hardware already possesses the necessary offsets for the presynaptic, postsynaptic, and subsequent layer. For the initial two layers, these parameters are stored in the Register File for immediate access upon start-up.

- **Input Spikes**: A dedicated region at the end of the layer blocks stores the incoming spike addresses. These are retrieved during cycles without memory contention from postsynaptic updates.

The number of elements per memory line is strictly constrained to be a **power of 2**. This design choice avoids the need for area-intensive integer division circuits, allowing the hardware to calculate memory offsets using efficient **logical bit-shift operations**. If the total number of weights in a layer is not a multiple of the packing factor, the remaining bits in the memory line are padded with zeros.

To avoid the significant computational and area overhead associated with floating-point units, the adLIF equations are implemented using **fixed-point arithmetic** [33]. The architecture utilizes an 8-bit format for weights and neuronal parameters, while a 12-bit format is reserved for internal states (synaptic currents, membrane potential, and adaptation current) to provide sufficient head-room against overflow. This precision remains fully configurable via compiler macros, allowing for a tailored trade-off between classification accuracy and hardware resource utilization.

2.2 Processing Element Design

The PE illustrated in Figure 2.6 is designed as a resource-shared architecture to minimize the hardware footprint and achieves the flexibility to execute any adLIF network. The unit is partitioned into three specialized functional modules:

- **Address Calculation Unit** : This module manages the pointer logic for SRAM navigation. It computes memory offsets for neuron states

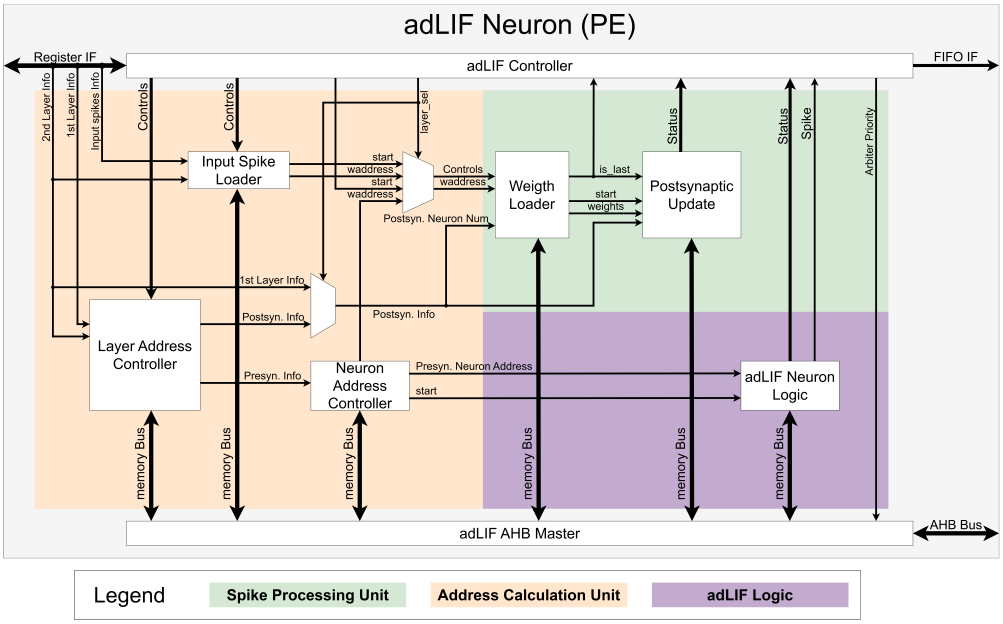

Figure 2.6: Internal architecture of the adLIF PE). The unit is partitioned into three main areas: the Address Calculation Unit (orange) for memory pointer management, the adLIF Logic (purple) for state updates, and the Spike Processing Unit (green) for weight integration. An adLIF Controller coordinates the execution, while the adLIF AHB Master manages communications with the system SRAM.

and synaptic weights in parallel with the neural computation, utilizing dedicated small-scale adders and multipliers.

- **adLIF Logic** : This module serves as the arithmetic backend responsible for updating the state variables $u[t]$ and $w[t]$. It utilizes a single pipelined multiplier [34] and an adder unit to execute the discretized adLIF equations derived in Chapter 1.

- **Spike Processing Unit** : This unit is triggered upon the generation or reception of a spike. It manages the fan-out phase by fetching synaptic weights and updating the accumulated synaptic currents (I_{syn}) for the neurons in the subsequent layer.

In addition to these functional units, the PE integrates two fundamental management blocks. An internal arbiter and interface, the **adLIF AHB Master**, handles concurrent memory requests from the internal modules and manages the communication protocol with the system SRAM. This block is

responsible for converting internal control signals (e.g. `wren`, `readen`) into standard AHB protocol transactions.

The entire execution flow is governed by the **adLIF Controller**, which orchestrates the set of FSMs. This controller also implements a priority signaling mechanism for the arbiter. This ensures that critical tasks, such as memory write-backs of state variables, are resolved efficiently without stalling the pipeline for low-priority requests.

Layer-Specific Operation and Data Flow

The PE adapts its operational flow according to the specific role of the layer currently residing in the execution pipeline:

- **Input Layer** : The PE operates in a purely **event-driven mode**. It translates incoming spike addresses into memory pointers to fetch the corresponding synaptic weights. The **Spike Processing Unit** then propagates these values to update the synaptic currents (I_{syn}) for the first hidden layer. In this stage, the adLIF Logic remains idle, as input nodes do not possess internal state variables.

- **Hidden Layers** : Each virtual neuron is updated sequentially according to the TDM model. The PE fetches the membrane potential, adaptation current, and neuronal parameters to execute the state update. If the threshold condition $u[t] \geq \vartheta$ is met, a spike is generated: the unit identifies the postsynaptic targets and updates their synaptic current buffers before proceeding to the next neuron index.

- **Output Layer** : The logic mirrors that of the hidden layers, with the exception that detected spikes are directly pushed to the Output FIFO for external classification. Since this represents the final processing stage, the **synaptic update phase is bypassed**, significantly reducing the computational latency of the final layer.

Datapath and adLIF Update Logic

The adLIF logic executes the discretized dynamics using a **shared fixed-point datapath**. The discrete-time equation for the two states variable, membrane potential (u) and adaptation current (w), introduced in Chapter 1, are solved sequentially for every neuron at each timestep:

$$\begin{cases} u[t] = \alpha u[t-1] + (1-\alpha)(I[t] - w[t-1]) \\ w[t] = \beta w[t-1] + (1-\beta)(a \cdot \hat{u}[t] + b \cdot S[t]) \end{cases} \tag{2.1}$$

In this equation, $I[t]$ represents the integrated synaptic input current, pre-calculated by the Spike Processing Unit through the accumulation of weights from active presynaptic neurons. The binary variable $S[t]$ indicates a spike event ($u \geq \vartheta$), while $\hat{u}[t]$ denotes the membrane potential after the soft reset mechanism is applied.

This architectural choice allows the design to reuse a single **pipelined multiplier** and adder unit across multiple calculation stages. Furthermore, this decomposition reduces the **critical path**, facilitating higher clock frequencies. The intermediate hardware operations are mapped to specific internal registers, which are exposed via the Debug Interface for cycle-accurate verification. This mapping, along with the sequence of operations, is detailed in Table 2.1.

Table 2.1: adLIF Processing Element: Parameter Mapping and Hardware Calculations. This table details the sequence of intermediate fixed-point operations and their corresponding debug register addresses.

Symbol	Formula	Description	Address
Input Parameters and State (Fetched from SRAM)			
I	–	Synaptic input current	`0x1F`
α, β	–	Decay factors	`0x21`, `0x24`
a, b	–	Coupling and adaptive offset coefficients	`0x25`, `0x28`
ϑ	–	Spike threshold	`0x23`
u, w	–	Previous state variables	`0x22`, `0x20`
Intermediate Calculations			
I_{adapt}	$I - w$	Adapted input current	`0x0F`
$\hat{\alpha}, \hat{\beta}$	$1-\alpha, 1-\beta$	Decay complements	`0x10`, `0x11`
I_{scaled}	$\hat{\alpha} \cdot I_{\text{adapt}}$	Scaled integration term	`0x12`
α_u	$\alpha \cdot u$	Decayed membrane potential	`0x13`
$\bar{u}$	$\alpha_u + I_{\text{scaled}}$	Pre-threshold potential	`0x14`
β_w	$\beta \cdot w$	Decayed adaptive current	`0x15`
$au{+}b$	$a \cdot \hat{u} + b$	Adaptive increment term	`0x1A`
β_{scaled}	$\hat{\beta} \cdot (\cdot)$	Scaled adaptive increment*	`0x18`
Output Calculations			
$\hat{u}$	$\bar{u} - \vartheta$ *or* $\bar{u}$	Updated potential (soft reset)	`0x17`
w_{new}	$\beta_w + \beta_{\text{scaled}}$	Updated adaptive current	`0x19`

*The argument is $(a \cdot \hat{u} + b)$ upon spike detection, and $(a \cdot \bar{u})$ otherwise.

Interleaved Pipeline Scheduling

To maximize throughput and reduce the total clock cycles per update, the PE employs an interleaved scheduling strategy. The architecture exploits the pipeline depth of the multiplier by overlapping data dependencies with SRAM operations:

- **Latency Hiding** : While the Logic Engine is computing the partial products for the current neuron state, the Address Calculation Unit and the parameter loader, included in the adLIF Neuron Logic, initiates the pre-fetching of states and parameters for the subsequent virtual neuron.
- **Functional Unit Utilization** : Whenever data dependencies permit, the adder and multiplier operate concurrently. For example,the multiplier calculates the potential decay $\alpha \cdot u$ while the adder prepares the adaptation decay $1 - \beta$.

By **masking memory access latency** and **overlapping arithmetic stages**, the PE achieves a full neuron state update in approximately 30–35 clock cycles. While this baseline is proportional to the number of virtual neurons, the actual execution time is slightly non-deterministic; the occurrence of a spike triggers additional logic for the fan-out phase, introducing minor variations in the cycle count. This **activity-dependent latency** is a fundamental characteristic of the event-driven nature of the architecture.

2.3 Design Trade-offs

The proposed TDM-based SNN architecture is designed to provide a flexible and scalable framework for neuromorphic event classification. However, the transition from a fully parallel spatial mapping to a serialized temporal mapping introduces several engineering trade-offs that impact area, latency, and power.

Memory vs Logic Usage

The use of a centralized SRAM to store layer informations allows the accelerator to support arbitrary **fully connected** topologies. Users can configure the number of layers and the neurons per layer at run-time without requiring a new hardware synthesis.

The primary constraint on scalability is not the logic area (LUTs/Flip-Flops or DSPs), which remains constant due to the resource-shared PE, but the available **SRAM capacity**. It is important to note that in almost all

neuromorphic or deep learning architectures, the storage of synaptic weights (M_w) represents a mandatory and dominant memory overhead [35]. The specific overhead introduced by our TDM virtualization is limited to the storage of virtual neuron states (M_n) and the layer control markers.

The storage requirements can be estimated as follows:

$$M_w = S_w \cdot \sum_{i=0}^{N_L-2} (n_i \cdot n_{i+1}) \tag{2.2}$$

$$M_n = S_p \cdot \sum_{i=1}^{N_L-1} n_i \tag{2.3}$$

Where S_w and S_p represent the storage size for a single weight (8 bits) and a single neuron state (12 bits for each of the 8 parameters), respectively. N_L is the total number of layers and n_i is the number of neurons in the i-th layer. For the larger PokerDVS network $2450 \times 32 \times 4$, the weights require approximately 76.7 KiB, while the states occupy only 0.42 KiB. This disparity is also due to the fact that input spiking neuron do not have an internal state to store. This confirms that synaptic density is the primary scaling bottleneck for this architecture.

Area vs Latency

The core compromise of this architecture lies in the balance between hardware footprint and inference speed. By reusing a single PE for all neurons, the design achieves an extremely low resource utilization, making it ideal for edge devices with limited area. The drawback of TDM is the **linear increase in latency** demonstrated in Chapter 4. Since neurons are updated sequentially, the total time required to complete a "Global Tick" sweep scales with the total neuron count. This sets a functional limit on the minimum temporal resolution (Δt) supported for a given network size. To mitigate this, the architecture utilizes interleaved scheduling and pipelining to mask memory access latency, allowing the PE to operate at higher clock frequencies and maximize throughput per cycle. While in all the architecture tested in this work our network stays well below the maximum amount of cycles, to further scale the system, a degree of parallelization need to be introduced to maintain low latency for larger topologies.

Power Consumption and Memory Traffic

Unlike fully parallel SNNs where local connections reduce data movement, this TDM architecture relies on a high-intensity memory traffic. Every neuron update requires a **Read-Modify-Write cycle** to the SRAM, which contributes

significantly to the dynamic power profile due to constant switching activity on the memory buses [36]. However, this power overhead is balanced by the efficiency of centralized logic. Large-scale parallel architectures often suffer from high **static leakage** and significant power dissipation within the routing required to interconnect distributed neurons [37, 38]. By concentrating the computation into a single, high-utilization PE, the system minimizes the number of active switching gates and simplifies the interconnect topology.

On the target Field Programmable Gate Array (FPGA) fabric, this approach reduces routing congestion and avoids the use of high-fanout global nets, which are often the primary bottlenecks for both timing closure and dynamic power. These architectural advantages would be further amplified in an Application-Specific Integrated Circuit (ASIC) implementation, where the reduction in total wire length and parasitic capacitance would lead to a more favorable energy-per-inference profile.

Furthermore, the architecture exploits the **temporal sparsity** of neuromorphic data. The input layer processing is strictly gated by incoming spikes, and the weight loader employs "row-skipping" logic to initiate memory transactions only for active neurons. Consequently, while the memory traffic per update is inherently high, the total power consumption remains optimized for edge applications where input signals are sparse and a low static power baseline is required.

Precision vs. Complexity

Finally, the choice of a parameterized fixed-point codification represents a trade-off between classification accuracy and hardware complexity [39]. While a higher bit-width provides a closer approximation to the continuous-time adLIF model, it significantly increases the area of the DSP slices and the width of the SRAM rows. As will be detailed in Chapter 3, the chosen 8-bit/12-bit precision was determined by evaluating the quantization error against a floating-point reference, ensuring that accuracy is preserved while maintaining a hardware-efficient datapath.

Chapter 3

Model Training and Datasets

This chapter presents the methodology for the training and optimization of the Adaptive Leaky Integrate and Fire (adLIF) network. The discussion begins with an analysis of the PokerDVS and N-MNIST neuromorphic benchmarks, evaluating their temporal dynamics and event statistics. A temporal accumulation strategy is described, which serves to synchronize asynchronous event streams with the discrete-time operation of the hardware accelerator. The focus then shifts to the training configuration, which utilizes a Mean Square Error (MSE) loss function to implement an output rate encoding scheme. Finally, the chapter details the Post-Training Quantization (PTQ) process and the evaluation of classification accuracy drop due to quantization prior to Field Programmable Gate Array (FPGA) implementation.

3.1 Neuromorphic Datasets Characteristics

The validation of the proposed Spiking Neural Network (SNN) architecture is performed using two benchmark datasets specifically designed for neuromorphic vision: **PokerDVS** [40] and **N-MNIST** [41]. These datasets consist of asynchronous event streams captured by Dynamic Vision Sensor (DVS). Both are obtained via the **Tonic** library [42], a specialized Python framework for the acquisition and preprocessing of event-based vision data. The events are structured as tuples (x, y, t, p), representing spatial coordinates, a microsecond-precision timestamp, and polarity. To convert the tuple (x, y, p) into the actual address of the input neuron (A_{spk}) , a linear mapping is applied. Specifically, the spatial coordinates and polarity are flattened into a single index following the formula:

$$A_{spk} = x + (y \cdot X_{max}) + (p \cdot X_{max} \cdot Y_{max}) \tag{3.1}$$

Table 3.1: Technical Specifications of Neuromorphic Benchmarks. Comparison of input dimensionality, temporal characteristics, and primary validation objectives.

Feature	**PokerDVS**	**N-MNIST**
Input Resolution	$35 \times 35 \times 2$	$34 \times 34 \times 2$
Sample Duration	10 - 30 ms	$\sim$300 ms
Avg. Events Rate (EPS)	134,567	13,399
Peak Event Rate (EPS)	265,772	26,394
Total Samples	68	70,000
Main Challenge	Latency, Throughput	Accuracy, Stability

where X_{max} and Y_{max} represent the sensor's width and height.

Technical specifications and primary challenges of each dataset are summarized in Table 3.1. These benchmarks allow for an evaluation across different operational regimes: PokerDVS emphasizes **high-bandwidth throughput and minimal latency**, while N-MNIST evaluates accuracy in a **large-scale, noisy classification task**.

3.1.1 PokerDVS

The PokerDVS dataset consists of event streams generated by flipping a deck of playing cards in front of a "Fast DVS" sensor. This work utilizes the Tonic version with a spatial resolution of 35×35 pixels. Including dual polarities, the network input is configured as a $35 \times 35 \times 2$ grid (2,450 channels). As illustrated in Figure 3.1, the dataset consists of four classes representing poker pips: Clubs, Diamonds, Hearts, and Spades. The complete dataset used in this study comprises 68 samples. A defining characteristic of PokerDVS is its **temporal speed**. Individual samples typically have a duration ranging from only 10 ms to 30 ms. This high-speed nature is reflected in the event statistics, which exhibit a **peak rate of approximately 265,772 Events Per Second (EPS)** and an average of 134,567 events per sample.

This dataset serves as a primary **benchmark** for evaluating the accelerator's **throughput and latency** response under high-bandwidth event bursts. While the limited number of samples makes it a relatively straightforward task for classification, it remains an effective benchmark for **architectural validation**. The reduced dataset size facilitates rapid training and testing cycles, which were leveraged to verify the impact of various network topologies on the temporal performance of the hardware.

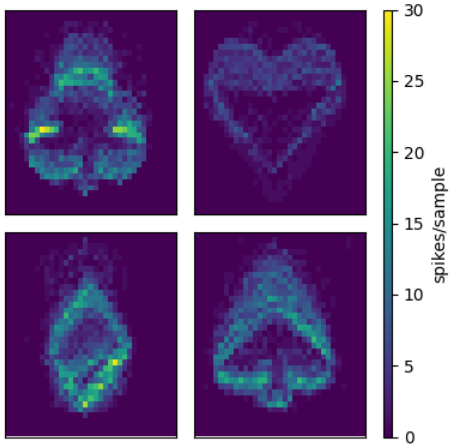

Figure 3.1: Visualization of PokerDVS Pip Classes. Integrated event frames for the four classes (Clubs, Diamonds, Hearts, Spades). The intensity of each pixel represents the accumulated spike count over the 10–30 ms sample duration.

3.1.2 N-MNIST (Neuromorphic MNIST)

The N-MNIST dataset is the neuromorphic equivalent of the static MNIST [43] handwritten digits. It was generated by mounting a DVS sensor on a pan-tilt unit and performing **three saccadic movements** (saccades) over the original images displayed on an LCD monitor. The spatial input resolution utilized in this implementation is 34×34 pixels. Including the dual polarity channels, this results in an input layer of **2,312 channels**. In contrast to PokerDVS, N-MNIST represents a **larger-scale classification challenge**, comprising 70,000 samples. Each recording has a significantly longer duration, approximately 300 ms, leading to a lower overall event density with an average rate of 13,399 EPS. A notable property of N-MNIST is the **periodicity of its event generation**: the peak event rate occurs near the midpoint of each of the three saccades, coinciding with the maximum rotational velocity of the sensor. While PokerDVS is primarily utilized to evaluate throughput and latency, N-MNIST challenges the network's stability and classification accuracy. The spatial overlap and **saccadic noise** inherent in the capturing process make certain digit classes particularly difficult to distinguish, such as the digits '9' and '5' illustrated in Figure 3.2.

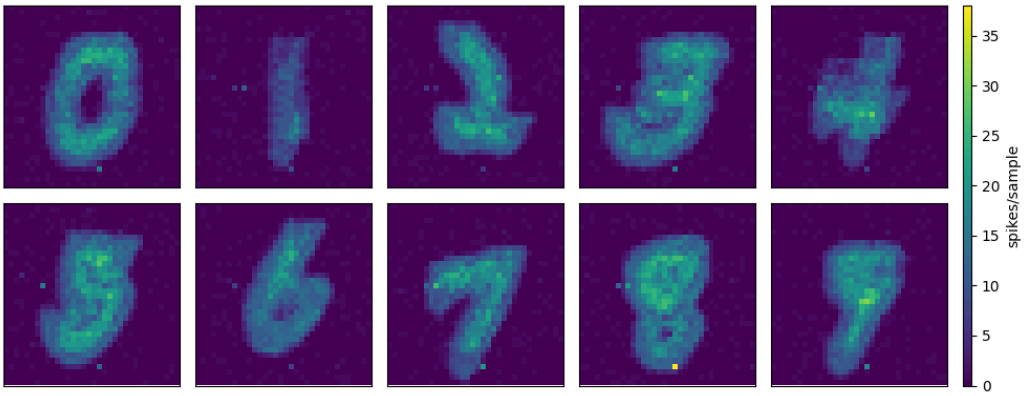

Figure 3.2: N-MNIST Digit Samples. Visualization of the ten digit classes (0-9). Each image represents the total spike count per pixel integrated over the entire sample duration ($\tilde{3}00$ ms). The visual "blur" is a direct result of the sensor's three saccadic movements over the static digit.

3.1.3 Statistical Analysis and Window Selection

To bridge the gap between the asynchronous nature of DVS event streams and the synchronous, discrete-time operation of the adLIF core, a preprocessing stage is required. This process, termed **temporal accumulation** or **time-binning**, transforms the continuous stream into a sequence of discrete time-steps (Δt). During this stage, all events (x, y, p) occurring within a specific window are aggregated into a single input frame, where Δt corresponds directly to the hardware's Global Tick period. Because this study utilizes pre-recorded datasets, this binning is performed as an **offline preprocessing stage** to prepare the data for training and hardware-accurate simulation.

The selection of the binning interval Δt is treated as a **hyperparameter**, a fixed configuration variable determined prior to the training of weights and neuron parameters to optimize system performance. It represents a critical architectural trade-off governed by two competing factors:

- **High Resolution (Small Δt)**: This setting preserves the fine temporal dynamics of the adLIF model, but increases the number of total steps to be processed, leading to higher inference latency. Since each time-step requires fixed computational cycles (e.g., for membrane decay and state updates), a very small Δt can lead to a high ratio of "empty" ticks, reducing the effective throughput of the Time Division Multiplexing (TDM) scheduler.

- **Low Resolution (Large Δt)**: Reduces the computational workload

Table 3.2: Comparison of DVS dataset characteristics and spike timing statistics. Summary of the temporal dynamics and the resulting parameters for training and hardware execution.

Property	**PokerDVS**	**N-MNIST**
Dataset Overview		
Avg. recording duration	17.40 ms	306.46 ms
Same-Coordinate Spikes		
Total Δt values	157,420	235,382,529
Median Δt	395 μs	4,449 μs
10th percentile Δt	116 μs	2,382 μs
Time Window Selection		
Selected time window	100 μs	2,000 μs
Avg. frames per sample	$\sim$174	$\sim$153

and training time but introduces "temporal blur". If Δt is too large, multiple spikes at the same coordinate may be collapsed into a single step, losing the advantage of SNNs in exploiting high-speed temporal information.

To ensure that Δt is matched to the physical characteristics of the input, the selection is not arbitrary but driven by the statistical distribution of the event data. This analysis focuses on the **inter-event intervals** at the **individual pixel** level rather than the global inter-event interval across the entire sensor array. The goal of this analysis was to define a Δt that minimizes temporal aliasing, the collision of multiple spikes at the same spatial coordinate within a single window. While the global spike rate describes raw sensor activity, the same-pixel interval defines the local frequency of the signal that the network must resolve. By matching Δt to this local distribution, the design avoids an explosion of "empty" frames that would occur with a global-scale resolution, while ensuring that the fastest relevant temporal changes are preserved for the adLIF dynamics.

Figure 3.3 illustrates the **probability density** of these intervals on a log-log scale, while Figure 3.4 presents the **Cumulative Distribution Function (CDF)**, which provides the mathematical basis for the window selection strategy. A quantitative comparison of the temporal characteristics and the resulting window selections for both benchmarks is presented in Table 3.2.

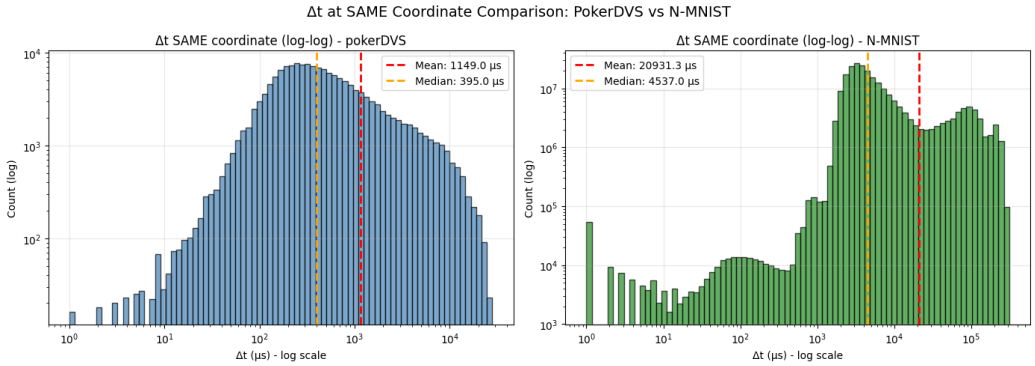

Figure 3.3: Inter-event interval (Δt) distribution for same-coordinate spikes. The plots use a log-log scale to highlight the distribution across several orders of magnitude. PokerDVS (left) shows a tighter distribution centered around $10^2 - 10^3 \mu s$, reflecting its high-speed nature. N-MNIST (right) exhibits a broader distribution with a significant peak near $10^3 - 10^4 \mu s$, consistent with slower saccadic sensor movements.

PokerDVS Analysis

As illustrated in Figure 3.3 and with the cumulative distribution in Figure 3.4, the analysis of $157,420$ same-coordinate inter-spike intervals reveals a distribution centered between 10^2 and 10^3 μs. The statistical distribution for these same-pixel spikes shows a 10^{th} **percentile** of 112 μs and a **median** of 392 μs.

Based on these findings, a **temporal window (Δt) of** 100 μs was selected for the hardware implementation. Since this value remains below the 10^{th} percentile of 112 μs recorded in the data, over 90% of consecutive spikes at any given spatial coordinate are guaranteed to be captured in separate temporal frames. This resolution yields approximately **170 frames** for an average 17.4 ms sample, providing a granular representation of the rapidly changing shapes of the poker pips.

N-MNIST Analysis

The N-MNIST dataset exhibits significantly **slower dynamics**, compared to PokerDVS. As illustrated in Figure 3.5, the event distribution is periodic, with three distinct activity peaks corresponding to the phases of maximum rotational velocity of the pan-tilt unit.

A statistical analysis of $235,382,529$ inter-event intervals at the same spatial coordinate reveals a distribution that is notably broader than that of PokerDVS, as shown in Figure 3.3. For this benchmark, the analysis indicates

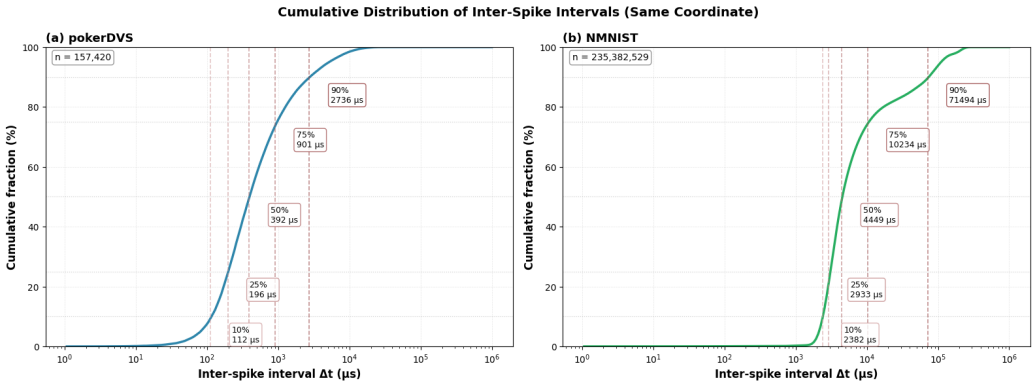

Figure 3.4: Cumulative Distribution Function (CDF) of Inter-event Intervals. This plot illustrates the percentage of events captured in separate windows as a function of the temporal window size. The 10^{th} percentile serves as the design target to ensure that the vast majority of consecutive spikes at the same spatial coordinate are processed in distinct temporal frames.

that the 10^{th} **percentile** for same-pixel spikes is $2,382\ \mu s$, while the **median** interval is $4,449\ \mu s$. These values, visualized in the cumulative distribution in Figure 3.4, confirm that the structural information of the handwritten digits is encoded over a significantly longer time scale.

Based on these findings, a **temporal window** (Δt) **of** $2\ ms$ was selected for the hardware implementation. This window ensures that more than 90% of consecutive spikes at the same pixel are captured in separate temporal frames, as the selected Δt remains below the $2,382\ \mu s$ threshold defined by the 10^{th} percentile. Given the average recording duration of approximately $306.46\ ms$, this choice reduces the total computational workload to roughly **150 inference cycles per sample**.

3.2 Network Topology and Training Strategy

The training and optimization of the network topologies were implemented using the snnTorch library [44], which facilitates gradient-based learning in spiking neural networks. The design of the network topology was driven by a **hardware-aware approach**, prioritizing a balance between classification performance and the physical constraints of the hardware memory footprint. This strategy involves two distinct validation paths: for **PokerDVS**, the focus is placed on assessing **temporal performance** across varying architectural complexities, while for **N-MNIST**, the objective is to **maximize accuracy** in a large-scale, high-noise classification task.

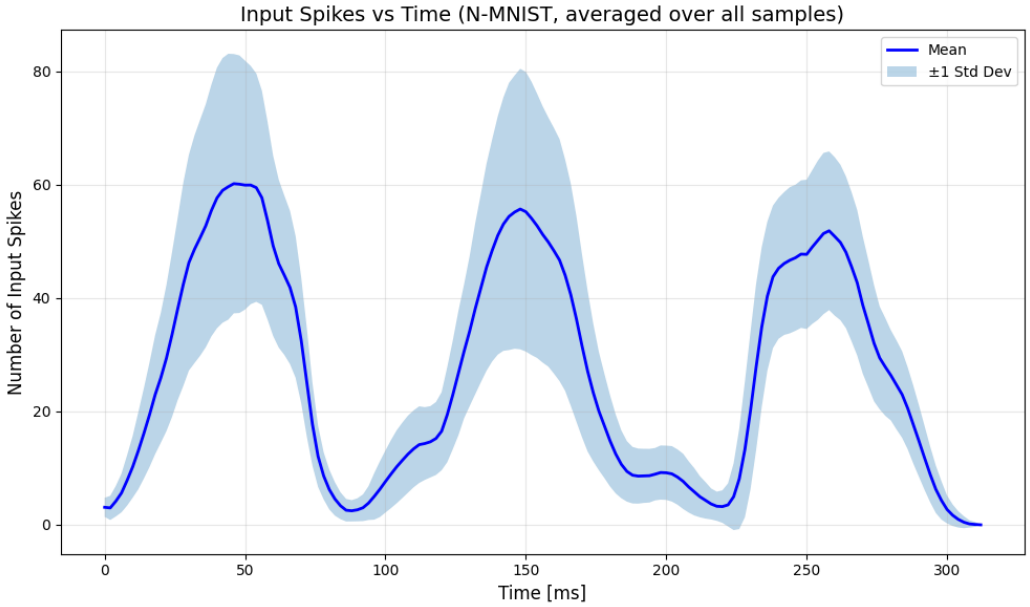

Figure 3.5: N-MNIST Event Distribution Over Time. The histogram illustrates the total spike count per 2 ms time-bin for a typical digit sample. The three distinct peaks correspond to the three saccadic movements of the DVS sensor.

Topology Selection and Hyperparameter Optimization

A primary challenge in training SNNs via backpropagation is the **non-differentiable nature** of the spiking threshold. In the adLIF model, the firing of a spike is determined by the **Heaviside step function** $H(U - \vartheta)$. The derivative of this function is zero everywhere except at the threshold, where it is undefined. This behavior prevents the flow of gradients during the backward pass, a phenomenon known as the **"dead neuron" problem** [45].

To address it, **Surrogate Gradient Descent** was utilized. During the forward pass, the neuron follows the standard discrete spiking logic, but during the backward pass, the non-differentiable derivative of the Heaviside function is replaced by a smooth, continuous approximation. In this work, the **Fast Sigmoid** (σ) surrogate function was selected [46]. The derivative used for backpropagation is defined as:

$$\sigma \approx \frac{u - \vartheta}{1 + k|u - \vartheta|} \implies \frac{\partial \sigma}{\partial u} \approx \frac{1}{(k|u - \vartheta| + 1)^2} \tag{3.2}$$

where u is the membrane potential, ϑ is the threshold, and k is the slope hyperparameter that modulates the sharpness of the gradient approximation.

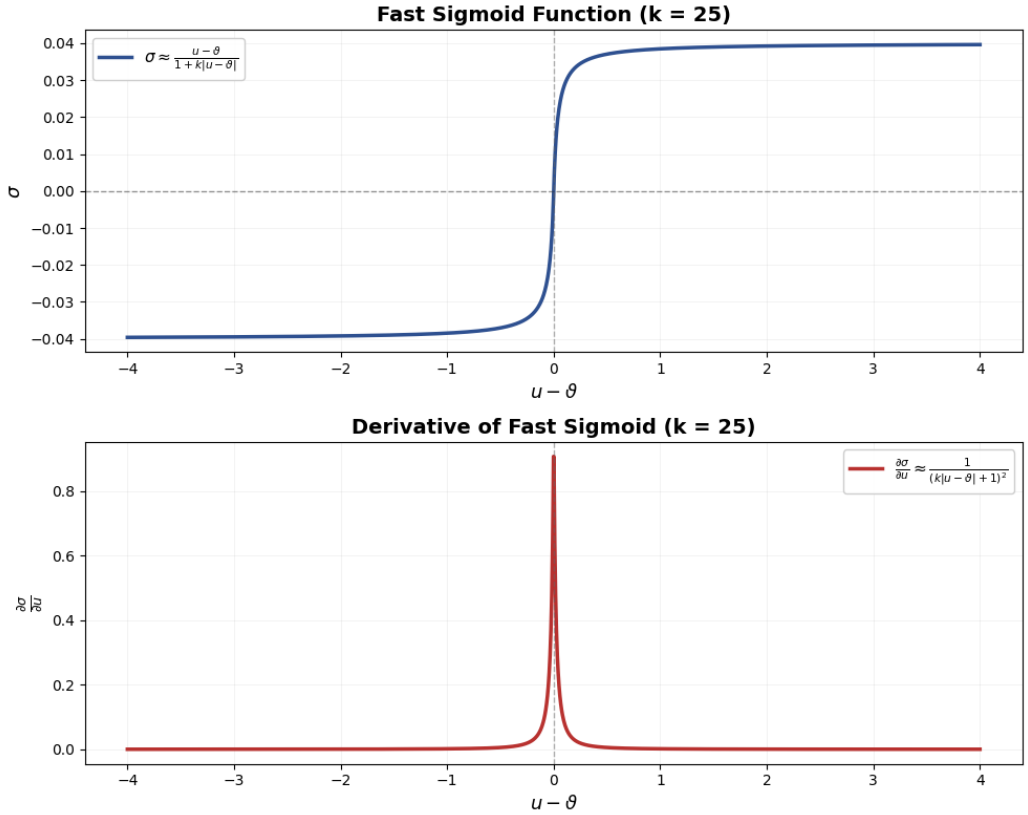

Figure 3.6: Fast Sigmoid Surrogate Function and its Derivative ($k = 25$). The top plot illustrates the surrogate function σ used to approximate the step-like firing behavior of the neuron. The bottom plot shows the corresponding derivative $\frac{\partial\sigma}{\partial u}$, which provides a smooth, non-zero gradient centered at the firing threshold ($u - \vartheta = 0$).

A slope of $k = 25$ was utilized for all training sessions. The Fast Sigmoid was chosen for its computational efficiency and its ability to provide a stable gradient signal over a wider range of membrane potentials compared to a standard sigmoid. Figure 3.6 represents the function and its derivatives.

To evaluate the scalability of the TDM architecture, **PokerDVS** was utilized to test **three distinct topologies**: a single-layer network (2450×4) and two multi-layer configurations ($2450 \times 16 \times 4$ and $2450 \times 32 \times 4$). In contrast, N-MNIST was trained using a single-layer 2312×10 topology, a choice driven by a Grid Search analysis that optimized batch sizes and loss function parameters. For both datasets, the intrinsic **adLIF parameters** ($\alpha, \beta, a, b, \vartheta$) were treated as **learnable variables** rather than static constants, enabling the network to adapt its internal dynamics to the specific temporal

Table 3.3: Network Configurations and Training Parameters.

Feature	PokerDVS (Small/Med./Large)	N-MNIST
Topology	$2450 \times \{0/16/32\} \times 4$	2312×10
Train/Test Split	48 / 20	60000 / 10000
Number of Epochs	15	15
Batch (Train/Test)	8 / 1	128 / 32
Loss Function	MSE Count Loss	MSE Count Loss
Correct Rate Target	0.8	0.8
Incorrect Rate Target	0.2	0.2

features of each dataset. The network configurations and training parameters are summarized in Table 3.3.

Output Encoding and Spike Rate Regulation

The training process employs a **Mean Square Error (MSE) loss function** based on the **total spike count** accumulated over the simulation time. For each sample, the loss is calculated as:

$$L = \frac{1}{N}\sum_{i=1}^{N}(S_i - \hat{S}_i)^2 \tag{3.3}$$

where N is the number of neurons in the output layer, S_i is the actual number of spikes emitted by neuron i during the simulation, and $\hat{S}_i$ is the target spike count. The target $\hat{S}_i$ is derived from the class label and defined as a percentage of the total simulation time-steps:

- For **correct class**: $\hat{S}_i = \texttt{num\_steps} \times \texttt{correct\_rate}$
- For **incorrect classes**: $\hat{S}_i = \texttt{num\_steps} \times \texttt{incorrect\_rate}$

This regulation ensures the network learns to discriminate classes via **Output Rate Encoding**. This approach was selected over temporal encoding (e.g., first-to-spike) due to its **superior robustness to input noise** and the generation of more stable training gradients.

The specific targets for the correct and incorrect firing rates were determined through empirical analysis. For the final network configurations, a correct rate of 0.8 and an incorrect rate of 0.2 were selected. These values were found to provide the optimal balance for the adLIF dynamics; setting the target rate too high introduces overstimulation noise that degrades classification

accuracy. Conversely, the incorrect rate was maintained at 0.2 rather than 0.0 to prevent the "dying neuron". By allowing a baseline level of activity even for incorrect classes, the optimizer maintains **non-zero gradients**, preventing neuronal parameters from settling into a permanent inactive state that would effectively remove them from the learning process.

3.3 Quantization Strategy

A Post Training Quantization (PTQ) strategy, supported by a **bit-accurate Golden Model**, was implemented to move from the high-level neural network training to the digital hardware implementation. This approach ensures that the mathematical behavior defined in the software environment is maintained in the fixed-point architecture of the accelerator, allowing for the evaluation of **accuracy degradation** prior to FPGA deployment. Initially, the network is **trained using 32-bit floating-point (FP32)** precision to ensure stable convergence of the learnable adLIF parameters. Following training, the synaptic weights and intrinsic neuronal parameters ($\alpha, \beta, a, b, \vartheta$) are converted into a fixed-point representation.

3.3.1 Bit-Accurate Golden Model

A dedicated **"Fixed-Point" neuron class** was developed in the software framework to validate the hardware design. This Golden Model emulates the behavior of the SystemVerilog implementation by replicating its arithmetic operations at each time-step. Every operation (multiplications, additions, and subtractions) is followed by a function to convert the result to fixed point, ensuring the values match the word lengths used in the hardware.

Division by $2^{N_{frac}}$ is implemented to simulate the right-shifting of partial products, maintaining the alignment of the decimal point. The model exposes internal variables, such as the scaled membrane potential and partial products of the adaptation current, which serve as reference points for cycle-accurate debugging. By comparing the outputs of this Golden Model with signals extracted from Register Transfer Level (RTL) simulations, a one-to-one match between the software's mathematical behavior and the hardware's execution was verified. This flow ensures that any degradation in performance is attributed to the quantization process rather than implementation errors in the digital logic.

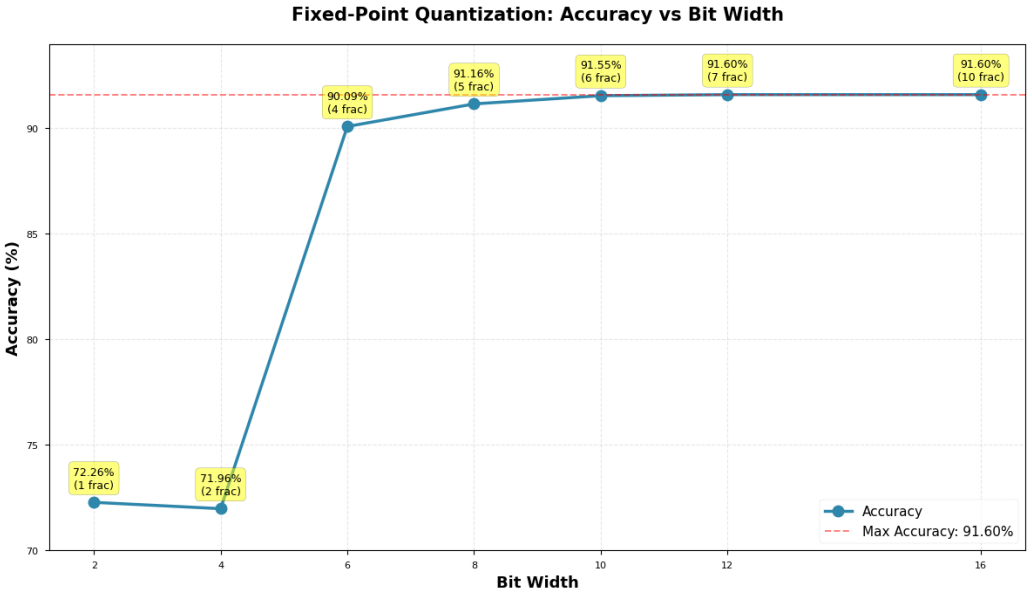

Figure 3.7: Classification accuracy as a function of bit-width quantization on the N-MNIST dataset. A sharp performance increase is observed between 4 bits and 8 bits. The accuracy was evaluated on the N-MNIST dataset in a Python environment.

3.3.2 Selection of Bit-Width and State Resolution

The selection of the target bit-width was based on an analysis of classification accuracy on the N-MNIST dataset across various quantization levels. As illustrated in Figure 3.7, accuracy remains stable above 8 bits but degrades when parameters are reduced to less than 6 bits.

An **8-bit parameter format with 5 fractional bits** ($N_{frac} = 5$) was selected as the trade-off between hardware efficiency and numerical precision. While 10-bit quantization offers a marginal improvement in accuracy (+0.39%), it increases hardware logic and memory requirements. The 8-bit format is byte-aligned, which simplifies the addressing logic and the operations required to extract parameters from memory.

While weights and parameters are capped at 8 bits, 12 bits are assigned to the synaptic current (I), membrane potential (u), and adaptive current (w). This resulting range of $[-64, 63.96875]$ is used to handle high-activity event bursts, where synaptic currents in the hidden layers can frequently saturate. The memory overhead for this expansion is negligible: for the $2450 \times 32 \times 4$ network, this expansion requires 432 additional bits compared to an all-8-bit implementation.

Chapter 4

Experimental Results

This chapter presents the experimental evaluation and hardware characterization of the proposed Adaptive Leaky Integrate and Fire (adLIF) accelerator.

The discussion begins with a description of the host-driven validation methodology, which utilizes a direct spike-injection via a micro-USB interface to run the test set on the FPGA. The focus then shifts to the hardware implementation on the AMD Artix-7 FPGA, providing an analysis of hardware resource utilization and evaluating the classification accuracy using the N-MNIST and PokerDVS benchmarks. A temporal performance study characterizes the linear scaling of processing latency relative to the number of input events, establishing the real-time operational boundaries for various network configurations. Finally, the architecture is compared against state-of-the-art FPGA-based SNN implementations, evaluating its efficiency in terms of logic area, power consumption, and accuracy.

4.1 Experimental Setup and Target Platform

The performance and efficiency of the proposed adLIF accelerator are evaluated through a series of experiments focusing on resource utilization, timing closure, and inference latency. This methodology demonstrate how the Time Division Multiplexing (TDM) architecture scales with varying neuromorphic workloads.

4.1.1 Validation Methodology and Test Flow

While the architecture supports real-time event acquisition via the Address Event Representation (AER) protocol, the validation phase utilizes a host-controlled environment to simplify the physical interface between the Field

Figure 4.1: Experimental Validation Setup. The FPGA development board is connected via UART to the host PC, where a Python environment performs real-time data injection and results analysis.

Programmable Gate Array (FPGA) and the workstation. By using the onboard micro-USB connection, the system utilizes the existing **UART-to-USB bridge** for data transfer and control, avoiding the additional hardware complexity of connecting the host PC to the FPGA's General-Purpose Input/Output (GPIO) pins.

Due to the bandwidth limitations of this UART-AHB bridge, testing is conducted using a **directed spike-injection method**. In this mode, the host PC explicitly controls the network's time-stepping, triggering the next execution cycle once the required data is fully loaded. This approach ensures that the communication overhead of the physical interface is completely decoupled from the internal hardware logic, allowing for an accurate evaluation of the accelerator's true computational latency.

The testing flow follows this approach:

1. **Network Deployment**: Following offline training, neuronal parameters and synaptic weights are quantized to the fixed-point formats justified in Section 3.3. These values are converted into memory-image files and loaded into the internal Static Random-Access Memory (SRAM) via the UART-AHB bridge.

2. **Dataset Pre-processing**: The test datasets are time-binned using a timestep Δt identical to the training phase. Events occurring within the window $[t_i, t_i + \Delta t]$ are grouped into frames consisting of the addresses of input neurons that spiked during that specific interval.

3. **Host-FPGA Synchronization**: A Python-based control script manages the communication, initializing the FPGA by writing the network configuration to the Register File and internal SRAM.

4. **Spike Injection**: For each frame, the host writes the list of event addresses directly into the system memory. Execution is triggered by writing to the `START_SNN` control register.

5. **Execution Monitoring**: The host monitors the `END_OF_TS` status flag. Once the Processing Element (PE) signals completion, the host retrieves output spikes from the Output First In First Out (FIFO) and reads the performance registers to log the total clock cycles consumed.

6. **Temporal Loop and State Reset**: This process iterates until all frames of a sample are processed. Between samples, internal neuron states (membrane potentials and adaptive currents) are reset to their initial values to ensure inference independence.

7. **Data Collection**: The procedure is repeated for the full test sets (10,000 samples for N-MNIST and 20 samples for PokerDVS). The results are stored in CSV files for final accuracy and latency benchmarking.

By utilizing this host-driven validation flow, the timing of the execution remains transparent to the hardware logic. This allows for a bit-accurate comparison between the FPGA implementation and the software-based Golden Model, regardless of the physical communication speed of the UART interface. Figure 4.1 illustrates the experimental setup, featuring the development board interfaced with a host PC, which manages commands and data flow through a Python environment.

4.1.2 Target Platform and Hardware Implementation

The architecture is implemented in **SystemVerilog** and synthesized using the AMD Vivado 2023.2 toolchain. The target hardware platform is an **AMD Artix-7 FPGA** (specifically the `xc7a35tcpg236-1` device), integrated into a CMOD-A7 development board.

While the CMOD-A7 provides an external 512 kB SRAM, its byte-addressable interface introduces a significant latency overhead of six clock

Table 4.1: Hardware Synthesis and Architecture Parameters.

Category	Parameter	Value
Numerical Precision	Weight (S_w)	8-bit
	Neuron Parameter (S_p)	8-bit
	Internal State (u, w, I)	12-bit
Memory Width	SRAM Word	48-bit
	SRAM Address	15-bit
	Input / Output FIFO Word	16-bit
	Input / Output FIFO Address	8-bit/4-bit
	AER Word	12-bit
Data Packing	Weights per Line (W_{pl})	4
	Currents per Line (I_{pl})	4
	Parameters per Line (P_{pl})	1

cycles per memory request for 48-bit words. Given that the TDM nature of our accelerator requires high-frequency access to neuron states and synaptic weights, such latency would severely bottleneck the PE. To evaluate the **true computational throughput** of the adLIF core and avoid external memory bottlenecks, the design utilizes **internal Block RAM (BRAM)** for primary storage.

Regarding the clocking infrastructure, the development board provides a 12 MHz reference clock. An internal high-speed **system clock of 100 MHz** is generated via an on-chip Mixed-Mode Clock Manager (MMCM). This clocking strategy provides the necessary temporal resolution for the adLIF dynamics while maintaining a reliable timing margin for the digital datapath.

The primary technical specifications and architecture parameters are summarized in Table 4.1. These values represent the configuration used for the final implementation and benchmarking on the target platform.

4.2 Resource Utilization and Physical Implementation

The modular design of the PE and the use of TDM result in an area-efficient footprint. Table 4.2 summarizes the hardware resources required for the complete system implementation on the target AMD Artix-7 FPGA.

The design occupies approximately **26.5% of the available Slice LUTs** (5,507 out of 20,800 available) and **11.2% of the Slice Registers** (4,668 out

Table 4.2: Hardware Resource Utilization on Xilinx Artix-7. Submodule totals may differ from parents due to cross-hierarchy optimization.

Module	LUTs	Flip Flops	DSPs	BRAM
Total System	**5,507**	**4,668**	**3**	**49.0**
UART-AHB Bridge	1,563	2,029	0	0
adLIF Accelerator	**3,945**	**2,629**	**3**	**49.0**
adLIF Top (PE Core)	**2,393**	**1,888**	**3**	**0**
Address Control	660	502	2	0
Neuron Logic	649	543	1	0
Spike Processing	310	406	0	0
*Other PE Logic**	779	437	0	0
Register File	430	435	0	0
SRAM Controller	0	0	0	48.0
Input/Output FIFOs	121	184	0	1.0
*Other Logic***	965	122	0	0

*Includes controller FSM , AHB Master and Debug registers.

**Includes Arbiter , Tick Gen , and Input Spike FSM.

of 41,600). By migrating the system SRAM and FIFOs to dedicated memory blocks, the implementation utilizes **49 Block RAM (BRAM) tiles**. This total consists of 48 RAMB36 macros and 2 RAMB18 macros, with the latter counting as 0.5 tiles each in the utilization report.

The **core accelerator logic** (`adLIF top`) consumes **only 3,945 LUTs**. The serialized nature of the TDM approach is highlighted by the requirement of only **3 DSP48E1 blocks** (3.33% utilization) to handle the adLIF arithmetic updates. Within this configuration, two DSP units are dedicated to the Address Control Unit for memory pointer arithmeticcalculating layer offsets based on neuron counts and determining synaptic weight addresses for input spikes. The third unit is utilized as the multiplier within the Neuron Logic block to compute the model equations presented in Chapter 1.

Notably, a significant portion of the prototype's resources is consumed by the UART-AHB Bridge (1,563 LUTs and 2,029 FFs). While necessary for host-PC communication in this implementation, this module is external to the accelerator core. In an integrated **System on chip (SoC)** environment using a **direct bus interface**, the logic footprint could be further reduced by approximately **30% for LUTs and 40% for Flip Flops**.

The system successfully meets all timing requirements at a **clock fre-**

Table 4.3: Memory Footprint per Network Topology. Comparison of the total memory lines, line width, and cumulative storage requirements in kilobytes for the evaluated SNN configurations.

Network Topology	Mem. Lines	Line Width	Storage (kB)
2450×4	2,480	48 bits	14.5
$2450 \times 16 \times 4$	9,963	48 bits	58.4
$2450 \times 32 \times 4$	19,895	48 bits	116.6
2312×10	7,010	48 bits	41.1

quency of 100 MHz, with a measured **Worst Negative Slack (WNS) of 0.111 ns**. Analysis of the **critical path** reveals that delay is primarily **dominated by routing (84%)**, a consequence of the centralized SRAM controller managing high-volume requests from a single PE. While additional pipeline stages on the memory interface could improve timing margins, the current performance already satisfies the **real-time latency requirements** for both PokerDVS and N-MNIST.

The memory utilization for the different configurations is summarized in Table 4.3. The total storage requirement scales according to the network depth and neuron number. For the most complex configuration utilized for **PokerDVS** ($2450 \times 32 \times 4$), the total footprint occupies **116.6 kB** of the internal BRAM.

4.3 Classification Accuracy and Quantization Impact

The adLIF accelerator was evaluated using two distinct neuromorphic benchmarks: PokerDVS and N-MNIST. These tests demonstrate the network's capability to handle different levels of temporal complexity and input activity on the Artix-7 target platform. In Figure 4.2, the performance of the different topology are presented, compared also with the pre-quantization ones.

PokerDVS Results

The PokerDVS dataset was used to validate the core functionality of the architecture across multiple topologies. While the **100% accuracy** achieved by the $2450 \times 32 \times 4$ and $2450 \times 16 \times 4$ networks configurations suggests a degree of **overfitting** due to the limited size of the Tonic dataset, this was not a primary concern for the scope of this study. The dataset served as a

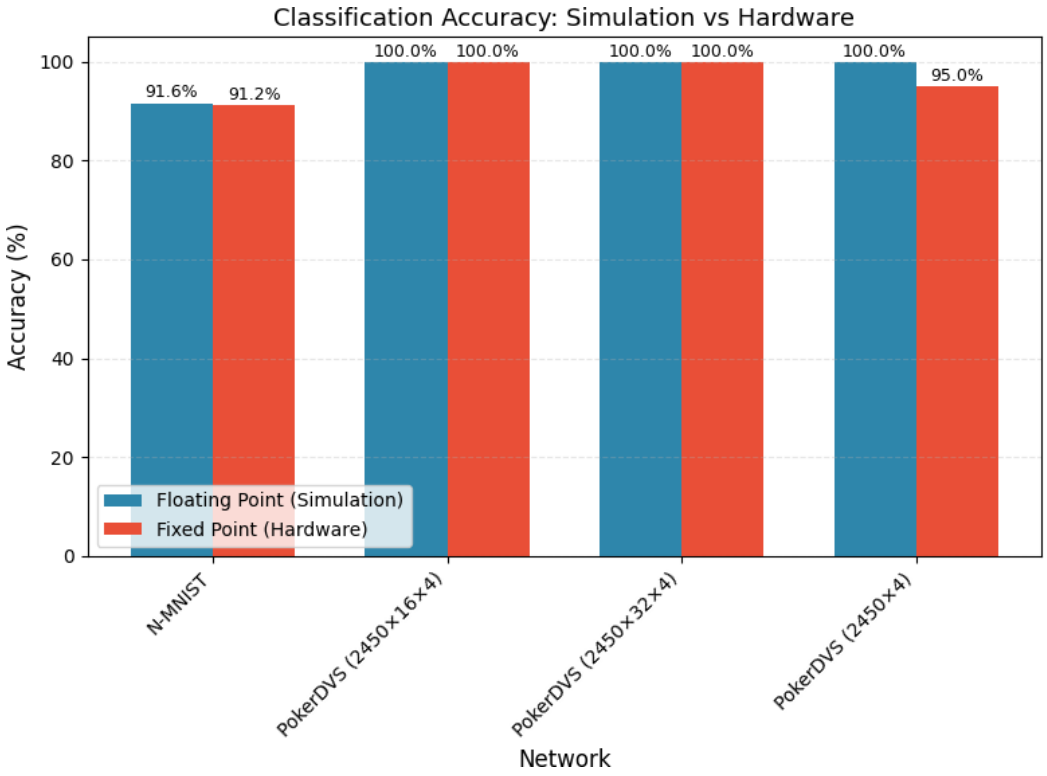

Figure 4.2: Classification Accuracy Across Multiple Topologies: FP32 vs. Fixed-Point. Comparison between the floating-point reference and fixed-point implementation for N-MNIST (2312×10) and three PokerDVS configurations ($2450 \times 16 \times 4$, $2450 \times 32 \times 4$, and 2450×4). While N-MNIST shows a minor degradation of 0.2% (91.6% to 91.4%), the multi-layer PokerDVS networks maintain perfect 100% accuracy. A slightly larger drop is observed in the single-layer PokerDVS baseline (95% vs 100%).

functional benchmark to evaluate **timing performances** and **architectural scalability** under high-speed event bursts, rather than to thoroughly test the network's generalization capabilities on this specific task.

The robustness of these multi-layer configurations indicates that the 8-bit parameter quantization does not compromise the temporal features inherent in the poker pips. The only observed discrepancy occurs in the smallest baseline network (2450×4), where the hardware implementation recorded a **single misclassification**, mistaking a "Heart" (sample 14) for a "Club". This specific error accounts for the drop to 95.0% accuracy seen in Figure 4.2. Given the near-perfect classification performance across the dataset, confusion matrices for the multi-layer networks provide no additional analytical insight and are thus omitted.

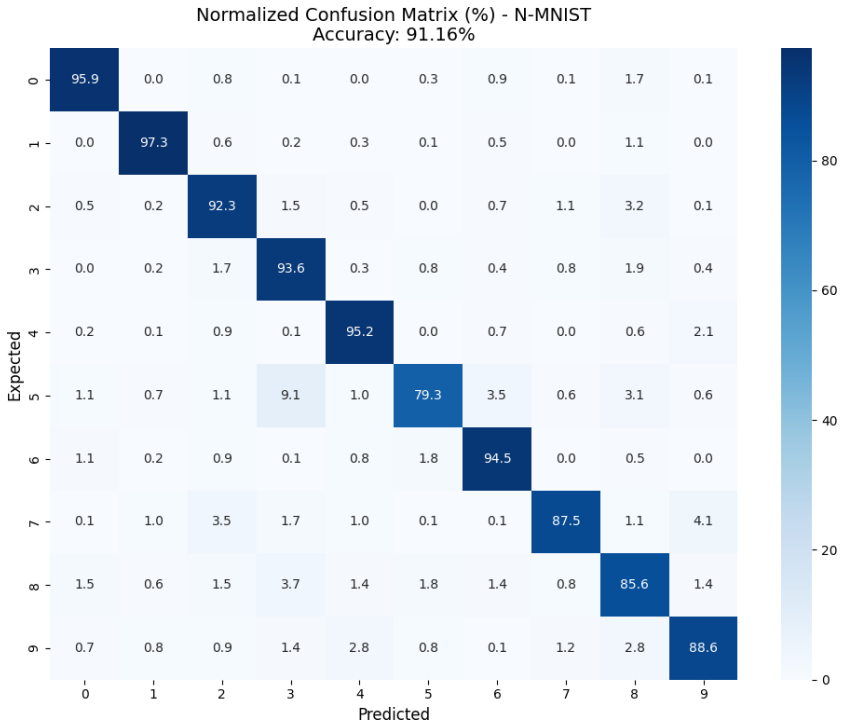

Figure 4.3: N-MNIST Normalized Confusion Matrix. Classification performance of the quantized model. The most significant confusion occurs in Class 5 (79.3 % accuracy), where 9.1 % of samples are misclassified as Class 3, likely due to shared spatial features blurred by the saccadic movement of the DVS sensor.

N-MNIST Results

The N-MNIST dataset presents a more rigorous challenge for the quantized datapath due to its extensive sample size and the presence of saccadic noise. The accelerator achieved a **classification accuracy of 91.16%**, a result that remains competitive within the state-of-the-art for FPGA-based SNNs.

The **normalized confusion matrix** in Figure 4.3 reveals specific error patterns linked to the neuromorphic nature of the data. While high-contrast digits such as **'0' and '1'** exhibit remarkable robustness, achieving **95.9 % and 97.3 %** correct predictions respectively, other classes show greater sensitivity to spatial blurring:

- **Class 5**: This class exhibits the highest misclassification rate in the hardware implementation, with an accuracy of 79.3 %. As indicated in the confusion matrix, there is significant overlap with Class 3 (9.1 % instances), Class 6 (3.5 %), and Class 8 (3.1 %). This low performance

is likely due to shared curvilinear spatial features that become indistinguishable during the peak rotational velocity of the DVS sensor's saccadic movements.

- **Class 8 and 7**: These digits show notable "leakage" into other categories. Class 8 (85.6%) is frequently confused with Class 3 (3.7%), while Class 7 (87.5%) is often misclassified as Class 9 (4.1%) or Class 2 (3.5%). This is a characteristic challenge in neuromorphic MNIST benchmarks, where the fast temporal evolution of loops and intersecting strokes can result in similar spike densities across the output layer adLIF neurons.

The overall stability of the results, despite the quantization of the adLIF parameters, confirms that the hardware core successfully replicates the dynamics required for digit recognition.

4.4 Temporal Performance and Latency Scaling

In this architecture, the **latency** required to process a single temporal window (Δt) is not fixed but **scales with the input activity** and the **network complexity**. To quantify this behavior, an analysis was performed by measuring the total clock cycles required for a time-step update across different topologies and datasets.

4.4.1 Linear Fit and Computational Cost

As illustrated in Figure 4.4, the relationship between the number of input spikes in a frame and the total duration of a processing cycles exhibits an exceptionally high linear correlation ($R^2 > 0.99$ for all configurations). This linearity confirms that the PE operates with a **predictable computational cost per event**. The results of the linear regression are modeled as

$$C = L_{base} + \gamma \cdot S_{in} \tag{4.1}$$

where C represents the number of clock cycles to complete the time-step update, L_{base} the baseline intercept and γ is the number of cycles per spike. The fit values are summarized in Table 4.4. All these values are reported for clock cycle in order to remove the dependency on the operational frequency.

The **intercept** (L_{base}) represents the constant clock cycles spent on **state updates** (membrane potential decay and adaptation) for all non-input neurons. As expected, this value is proportional to the total number of virtual

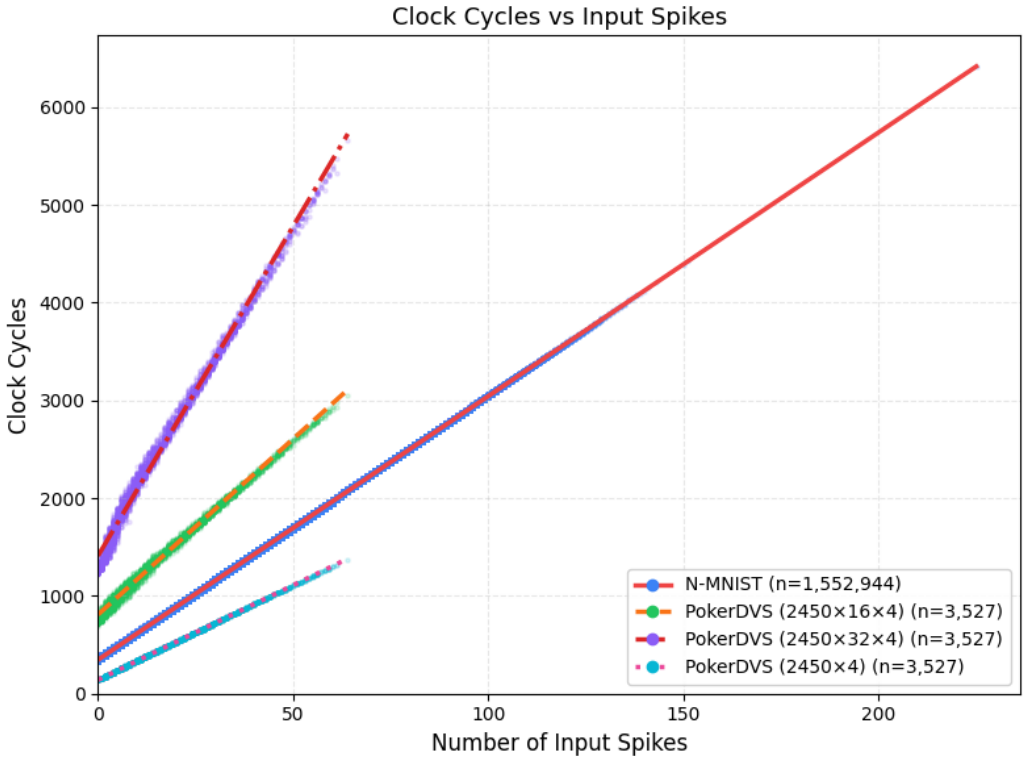

Figure 4.4: Linear Scaling of Computational Latency. Clock cycles per Global Tick as a function of input spikes. The high R^2 values across all topologies validate the TDM execution model's predictability.

neurons in the hidden and output layers, for which a state update is required at every timestep regardless of input activity.

Similarly, the slope (γ) is strongly dependent on the total post-synaptic fan-out. In a TDM-based SNN, each input spike triggers a sequential update of all synapses in the subsequent layers. This relationship is empirically confirmed by comparing the PokerDVS configurations:

- In the $2450 \times 16 \times 4$ topology, an input spike must potentially propagate to **20 non-input neurons** $(16 + 4)$, resulting in a slope $\gamma \approx 36.0$.
- In the $2450 \times 32 \times 4$ topology, the fan-out increases to **36 non-input neurons** $(32 + 4)$, with a corresponding slope $\gamma \approx 67.5$.

The **ratio between the marginal cost and the total fan-out** remains consistent across both configurations ($\gamma_{16}/20 \approx 1.80$ vs. $\gamma_{32}/36 \approx 1.87$), a result that is intuitively consistent with the underlying TDM execution model. While a broader range of topologies would be required to statistically formalize

Table 4.4: Linear Fit Parameters for Latency Analysis. The intercept (L_{base}) represents the fixed update cost, while the slope (γ) denotes the marginal cost per input event.

Topology	**Base Cycles (L_{base})**	**Cycles/Spike (γ)**	R^2
N-MNIST			
2312×10	346.46	26.97	0.9999
PokerDVS			
2450×4	141.12	19.38	0.9997
$2450 \times 16 \times 4$	809.34	35.95	0.9941
$2450 \times 32 \times 4$	1410.13	67.45	0.9922

this dependency, the current results already provide a strong indication that the **marginal cost per spike** is primarily dictated by the total **synaptic updates** the hardware must perform. This suggests that the **computational overhead scales predictably** with the network's architectural complexity.

4.4.2 Analysis of Deviations from Linearity

While the linear regression provides a robust model for predicting latency, a deeper inspection of the experimental data reveals specific second-order effects. These deviations from the ideal linear fit originate from both the hardware architecture's execution flow and the inherent dynamics of the neuron model.

- **Multi-Layer Stochasticity**: A wider distribution of data points is observed in the deeper topologies ($2450 \times 16 \times 4$ and $2450 \times 32 \times 4$) at low input spike counts. This variance originates from the probabilistic nature of hidden layer activations. At low activity levels, hidden neurons reach their firing thresholds inconsistently. However, when they fire, they trigger a "second wave" of synaptic updates in the subsequent output layer. This likely increases the total cycle count for those specific frames, causing them to deviate from the baseline linear prediction. In contrast, two-layer networks (e.g., N-MNIST 2312×10) exhibit a tighter linear fit, presumably because output spikes do not trigger further post-synaptic integration cycles.
- **Pipeline Amortization**: A subtle "dual-slope" trend is observable in three-layer topologies, where the marginal cost per spike (γ) seems to decrease during periods of very high activity. As illustrated in Figure

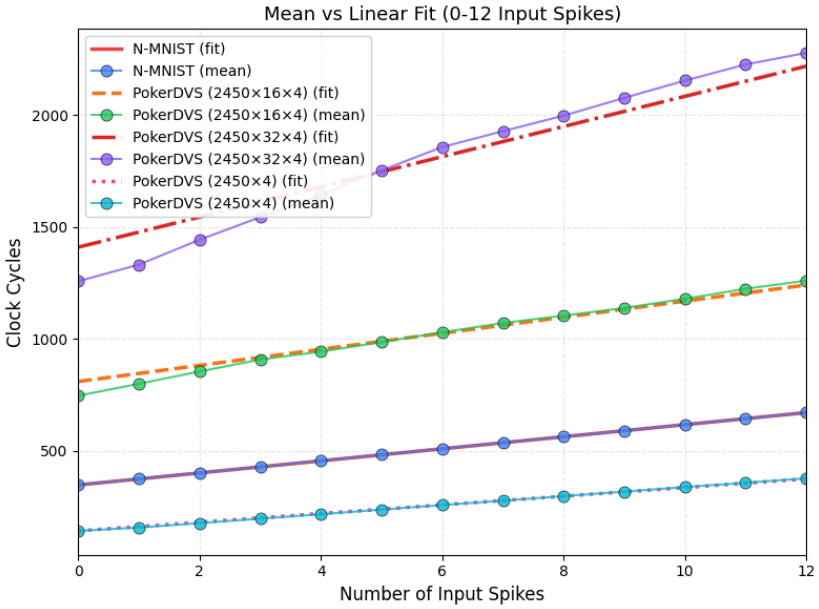

Figure 4.5: Zoom of Linear Fit. A zoomed-in view of the clock cycles per frame, comparing the calculated linear fit and the empirical mean. The divergence between the mean and the fit at low spike counts suggests a potential "dual-slope" behavior, likely caused by improved pipeline amortization and reduced state-transition overhead during high-activity bursts.

4.5, there is a slight divergence between the empirical average latency and the linear fit. This behavior could be attributed to increased pipeline occupancy within the PE. As the input spike count rises, the fixed hardware overhead associated with state-update transitions may be better amortized. Furthermore, it is possible that the hardware effectively hides memory access latencies by maintaining a continuous, uninterrupted flow of synaptic integrations during high-activity bursts, thereby increasing overall computational efficiency.

- **Inhibitory Weights and Firing Suppression**: The presence of inhibitory synapses (negative weights) serves to reduce the membrane potential of post-synaptic neurons. From a hardware perspective, processing an inhibitory weight consumes the same number of clock cycles as an excitatory one, as the Arithmetic Logic Unit (ALU) performs the same subtraction operation. However, by suppressing post-synaptic spikes, inhibitory activity indirectly reduces the total latency of subsequent layers by preventing the "second wave" of updates discussed previously.

- **Spontaneous Firing and Intercept Sensitivity**: In some configurations, such as the $2450 \times 32 \times 4$ network, some neurons may be programmed with negative thresholds ($\vartheta < 0$), leading to spontaneous firing even in the absence of input activity. These "spontaneous" events require the Processing Element to execute spike-reset logic and update adaptive currents (w) during the baseline sweep. This behavior accounts for part of the higher intercept (L_{base}) observed in more complex topologies, as the baseline cycle count must reflect not only the passive decay updates but also the management of internally generated spikes and their subsequent refractory resets.

4.4.3 Throughput and Real-Time Feasibility

To conclude the temporal analysis, the measured linear fit parameters are used to establish the operational boundaries of the accelerator. By projecting the maximum number of input spikes that the PE can handle within a single temporal window (Δt),the **maximum throughput** in terms of Events Per Second (EPS) is determined.

The maximum clock cycles available to process a single Global Tick, CC_{max}, is constrained by the duration of the time window Δt and the system clock period T_{clk} (10 ns for a 100 MHz clock). This limit is defined as:

$$CC_{max} = \frac{\Delta t}{T_{clk}} \tag{4.2}$$

By rearranging the linear model derived in Section 4.4.1, the **maximum number of input spikes** n_{spk_max} that can be processed before exceeding the **real-time deadline** is given by:

$$n_{spk\_max} = \frac{CC_{max} - L_{base}}{\gamma} \tag{4.3}$$

where L_{base} is the intercept (baseline overhead) and γ is the slope (cycles per spike). From this value, the **maximum sustainable event throughput** EPS_{max} for the architecture can be calculated as:

$$EPS_{max} = \frac{n_{spk\_max}}{\Delta t} \tag{4.4}$$

The calculated limits for each configuration are presented in Table 4.5. For N-MNIST, the longer Δt window allows for a higher number of spikes per timestep, while the tight 100 μs window of PokerDVS demonstrates the architecture's ability to handle **high-bandwidth bursts** in real-time conditions.

Table 4.5: Maximum Throughput and Real-Time Limits. The table shows the operational limits for each network topology at 100 MHz.

Topology	Δt (μs)	CC_{max}	n_{spk_max}	EPS_{max} **(MEPS)**
2312×10	2000	200,000	7,400	3.70
2450×4	100	10,000	508	5.08
$2450 \times 16 \times 4$	100	10,000	255	2.55
$2450 \times 32 \times 4$	100	10,000	127	1.27

This analysis confirms that even for the most complex configuration ($2450 \times 32 \times 4$), the system supports a **maximum event rate of 1.27 million events per second (MEPS)**. Given that the average activity recorded for the PokerDVS sensor is significantly lower (134,567), the TDM-based adLIF core provides a substantial **temporal safety margin**, ensuring reliable operation even during peak activity bursts.

4.5 Power Consumption Analysis

To evaluate the energy efficiency of the adLIF accelerator, a power analysis was conducted based on the post-routing implementation results. The estimation was performed using the **Vivado Report Power tool**. The target device for this analysis is the Xilinx Artix-7 (`xc7a35tcpg236-1`). The **total estimated on-chip power** for the system is **311 mW**. This total is composed of **236 mW of dynamic power and 75 mW of device static power**. The static component is largely determined by the transistor leakage of the FPGA fabric and remains constant regardless of the neural network activity.

A detailed breakdown of the dynamic power by component is provided in Table 4.6. The analysis reveals that the MMCM is the primary consumer of dynamic power, accounting for 104 mW, which represents approximately 44% of the total dynamic power budget. This high consumption is characteristic of FPGA clocking resources required to generate high-speed internal clocks from a lower-frequency external reference.

Beyond the clock generation, the **Block RAM (BRAM)** utilization is the **second most significant factor**, consuming 78 mW. This is consistent with the TDM architecture's high-frequency memory access patterns required to fetch weights and update neuron states at every Global Tick. The core processing logic, including Slice Logic and DSPs, consumes a relatively small fraction of the power ($\sim 15\ mW$).

The power consumption is heavily influenced by the underlying FPGA

Table 4.6: Dynamic Power Consumption Breakdown. Detailed on-chip power distribution for the implemented SNN system.

Component	Power (mW)	Percentage (%)
Clock Generation (MMCM)	104	44.07
Block RAM	78	33.05
Signals	21	8.90
Clocks (Distribution)	16	6.78
Slice Logic (LUT/Reg)	14	5.93
I/O and DSP	3	1.27
Total Dynamic	**236**	**100.00**

architecture. The overhead of the MMCM and the general-purpose routing fabric (Signals) accounts for a substantial portion of the energy budget. When transitioning from a programmable platform to an Application-Specific Integrated Circuit (ASIC) implementation, the power consumption is expected to decrease significantly. In an ASIC, clock trees can be optimized for specific frequencies and the dedicated datapath would eliminate the parasitic capacitance associated with general-purpose FPGA interconnects.

4.6 Comparison with State-of-the-Art

Spiking Neural Network (SNN) hardware implementations vary significantly in terms of area, power consumption, and throughput depending on their specific design objectives. In literature, architectures **differ across several dimensions**: the choice of **neuron model** (Leaky Integrate and Fire (LIF), adLIF, Izhikevich, etc.), the **degree of parallelism** (ranging from single-neuron TDM to fully physical parallel arrays), and the underlying **implementation platform** (synchronous, asynchronous, analog, or digital).

Given this extreme heterogeneity, this comparison focuses on architectures that share fundamental characteristics with the proposed work: **digital, synchronous FPGA implementations tested on image detection (MNIST or N-MNIST)**. Almost every architecture shares a TDM approach to reduce resources usage. TripleBrain [47] has been added because it uses the same neuromorphic dataset N-MNIST. While their architecture employs a large parallelization with an array of tiles, the processing inside each tile uses a TDM approach. The works of [29] and [48] utilize serialization strategies. While [29] incorporates a degree of parallelism alongside neuron-level TDM, [48] pushes the serialization to the extreme using a **bit-serial hardware**

Table 4.7: Comparison with State-of-the-Art FPGA SNN Accelerators. Performance benchmarks against architectures targeting similar neuromorphic or static MNIST datasets.

Metric	This Work	Wang et al. [47]	Valencia et al.[49]	Wu et al.[29]	Losh et al.[48]	Zhang et al.[51]
Neur. Model	adLIF	LIF	Izhikevich	LIF	Spike-Like (IF & ReLU)	IF
Learning	Offline	Online (STDP)	Offline	Online (STDP)	Offline	Offline
Topology	2312×10	784×256	1000×1000*	784×100	$196 \times 64 \times 10$	400×10
Reg./FFs	4,668	8,505	11,139	19,502	1,003	7,765
LUTs	5,507	10,052	12,700	51,007	1,335	5,897
DSPs	3	32	110	430	-	1
BRAM (tiles)	49	131	14.5	0	-	8 (+ 710 LUTRAM)
Speed (MHz)	100	250	92	303.4	125	100
Power (mW)	311	938	-	-	142	627
FPGA Device	Artix-7 (XC7A35T)	Zynq-7045	Artix-7 (XC7A200T)	Zynq-7035	Zynq-7010	Kintex XCKU035
Dataset	N-MNIST	N-MNIST	MNIST	MNIST	MNIST	MNIST
Accuracy	91.16 %	82.32 %	89 %	80.8 %	90.7 %	90.39 %

*While the resource usage refers to a 1000×1000 network, the accuracy results in [49] are reported for a smaller 784×100 network.

architecture, achieving **minimal resource** utilization at the expense of **increased latency**. Also [49] uses a form of TDM approach, which the authors call "**folding transformations**", where a set of parallel processing neurons operates in a TDM style.

For a broader perspective on the current landscape of neuromorphic hardware, the reader is referred to the comprehensive meta-review in [50], which evaluates a vast array of FPGA-based architectures. Table 4.7 summarizes the performance and resource utilization of the proposed adLIF accelerator against these selected implementations.

4.6.1 Accuracy and Dataset Complexity

While the majority of FPGA-based SNN implementations are benchmarked using the static MNIST dataset, this work utilizes N-MNIST, which presents a significantly more rigorous challenge due to the inclusion of saccadic noise and high-frequency temporal events. Although **classification accuracy** is primarily a **function of the network topology and the training algorithm** rather than the hardware itself, a comparison is necessary to **validate** that the proposed 12-bit/8-bit quantization strategy does not compromise the model's ability to represent complex spatiotemporal features.

The most direct comparison for this work is **TripleBrain** [47], which also targets the N-MNIST dataset. TripleBrain however utilizes local, on-chip learning rules as Spike-timing-dependent plasticity (STDP) [52, 53] and achieves a classification accuracy of 82.32%. In contrast, our architecture achieves 91.16%, representing an 8.8% improvement. This accuracy gap validates the **choice of an offline optimization strategy** (using surrogate gradients) over local on-chip rules. While local rules like STDP enable adaptability at the edge, they often struggle to converge to global optima

when processing the temporal dependencies of neuromorphic saccadic data.

The other works target the static MNIST dataset. While the classic MNIST dataset remains a frequent benchmark, it fails to exploit the inherently event-driven nature of spiking architectures. Processing static images typically requires arbitrary spike encoding schemes that may introduce bias into the performance evaluation. Neuromorphic architectures are fundamentally more efficient and effective when evaluated on natively temporal tasks, where the adLIF dynamics can be fully leveraged.

This comparison confirms that the proposed architecture provides the necessary numerical precision to remain competitive with state-of-the-art designs while offering the flexibility to handle the more complex temporal dynamics associated with Dynamic Vision Sensor (DVS)-based sensors.

4.6.2 Resource Efficiency

The proposed Single-PE TDM architecture demonstrates a **distinct advantage** in **minimizing hardware footprint** while supporting complex neuronal dynamics. This design consumes only 5,507 LUTs and 3 DSPs to support a network of over 2,300 neurons.

When compared to TripleBrain [47], which utilizes a parallel 2D array of Neural Processing Tiles (NPTs), our design occupies approximately 40% fewer LUTs while supporting a network with double the neuron count. While **TripleBrain's overhead** is mainly attributed to support for **on-chip learning**, our architecture demonstrates that a serialized TDM approach provides a significantly more scalable path for high-density inference at the edge.

Similarly, the architecture by Valencia et al. [49] employs a hybrid parallel-sequential scheme that requires 12,700 LUTs and 11,139 FFs. This comparison highlights that the serialized TDM approach adopted in this work drastically reduces logic area compared to other implementations. The efficiency gap is pronounced when comparing arithmetic resources (DSPs). Valencia et al. require 110 DSP slices largely because they implement the Izhikevich model, which involves computationally expensive quadratic terms ($0.04v^2 + 5v$). In contrast, by utilizing the linearized adLIF model, this work achieves temporal adaptation using only 3 DSPs (one for the neuron logic and two for address calculation), effectively avoiding the heavy arithmetic cost of quadratic solvers or CORDIC algorithms used by Wu et al[29] (which consume over 51k LUTs and 430 DSPs as noted in Table 4.7).

A similar amount of resources is consumed instead by Zhang et al [51] which consume way less BRAM but for a network with a reduced number of input neurons, by cropping the MNIST images to a 20x20 input grid.

The **only architecture with a smaller LUT footprint** is the bit-serial

design by Losh and Llamocca [48] (1,335 LUTs and 1003 Registers). However, their "Spike-Like" model relies on simple counters and comparators rather than biological membrane dynamics. While they implement a bias term to approximate leakage, the design trades off the high-precision temporal dynamics required for complex neuromorphic datasets like N-MNIST to achieve minimal area on static MNIST tasks.

Consequently, the proposed serialized TDM approach offers a superior balance, providing the high density of bit-serial designs with the dynamic complexity of DSP-rich architectures.

4.6.3 Power Consumption

The proposed architecture consumes a total of 311 mW (236 mW dynamic + 75 mW static). As shown in Table 4.7, this is significantly lower than the 938 mW consumed by TripleBrain [47]. The reduction relative to TripleBrain is attributed to the removal of complex on-chip learning logic. Furthermore, the reduction relative to other high-performance architectures is due to the elimination of high-power external memory interfaces, such as DDR3, by utilizing internal BRAM.

The efficiency of this design is further highlighted when compared to the TDM implementation by Zhang et al. [51]. Their optimized "Design 3" presented in the paper, which also utilizes resource reuse, consumes 627 mW to support a network topology of 400×10. In contrast, the proposed architecture achieves a $2\times$ reduction in power while supporting a network with more 5 times the amount of input neurons (2302 vs 400). While the bit-serial architecture by Losh and Llamocca [48] reports a lower total power of 142 mW it targets a significantly smaller network and utilizes a simplified "Spike-Like" model that lacks the temporal complexity of the adLIF dynamics. The proposed design therefore offers a superior power-to-complexity ratio, providing adaptive biological dynamics within a competitive power budget.

Furthermore, power analysis reveals that 53% of the dynamic power (104 mW) in this work is consumed by the MMCM (Mixed-Mode Clock Manager) to generate the 100 MHz system clock from the 12 MHz board reference. Total power consumption would likely decrease substantially in an ASIC implementation, where clock trees are specifically optimized and the high-power overhead of programmable FPGA clock managers is eliminated. This hypothesis is supported by Valencia et al. [49], who demonstrated that their generalized SNN architecture synthesized for 32-nm CMOS technology dissipates only 3.6 mW in ASIC, suggesting that the majority of the current consumption is attributable to the FPGA fabric and routing overhead rather than the computational logic of the SNN itself.

Chapter 5

Conclusion

The experimental evaluation on the AMD Artix-7 platform demonstrates that the proposed Adaptive Leaky Integrate and Fire (adLIF) accelerator achieves an **optimal balance** between biological complexity, classification accuracy, and hardware efficiency. The validation followed a dual-database approach to stress-test different architectural requirements: **PokerDVS** was utilized to evaluate the system's ability to **handle high-bandwidth**, low-latency event bursts, while **N-MNIST** served to validate the **stability and accuracy** of the 12-bit/8-bit quantized datapath in a large-scale classification task.

The implementation highlights include:

- **Resource Efficiency**: The complete system occupies only **5,507 LUTs** and **4,668 Flip-Flops**. The serialized nature of the architecture is emphasized by the requirement of just **3 DSP** slices to handle the model's arithmetic, while **49 BRAM** tiles provide the necessary storage for synaptic weights and neuronal states.

- **Accuracy** The quantization strategy preserves the complex spatiotemporal dynamics of the adLIF model, resulting in a classification accuracy of **91.16%** on the N-MNIST dataset. This represents the **highest accuracy** among the analyzed SNN architectures, even when compared to those evaluated on the simpler MNIST benchmark.

- **Temporal Performance**: The architecture exhibits a highly predictable **linear scaling of latency** ($R^2 > 0.99$), supporting a maximum event throughput of **5.08 MEPS** for the single-layer baseline and **1.27 MEPS** for the most complex $2450 \times 32 \times 4$ topology. This linearity provides a robust methodology to **estimate latency prior to implementation**, allowing for architectural adjustments based on the timing analysis of expected input data.

Table 5.1: Performance Summary of the adLIF Accelerator.

Parameter	Achieved Value
Target Device	AMD Artix-7 (XC7A35T)
System Clock	100 MHz
Logic Utilization (LUTs / FFs)	5,507 / 4,668
Arithmetic Resources (DSPs)	3
Memory Resources (BRAM)	49 tiles
Total On-Chip Power	311 mW
Dynamic / Static Power	236 mW / 75 mW
Max Throughput (EPS_{max})	1.27 – 5.08 MEPS
N-MNIST Accuracy	91.16%

- **Power**: The system operates with a total power consumption of **311 mW**, of which **236 mW is dynamic power**. Approximately **44% (104 mW)** of the dynamic portion is attributed to the **Mixed-Mode Clock Manager (MMCM) clock generation** required for **100 MHz operation** within the FPGA platform. Given the impact of the static power component, further efficiency gains could be achieved by transitioning to a different hardware platform.

Table 5.1 summarizes the main performance metrics achieved by the adLIF accelerator:

This work provides a viable hardware solution for low-power, high-accuracy processing of asynchronous sensor data. By utilizing the linearized adLIF model within a Time Division Multiplexing (TDM) framework, the accelerator achieves sophisticated adaptive dynamics with a logic density suitable for resource-constrained edge devices.

5.1 Future Works

The development of the **adLIF accelerator** can be extended in several directions to further improve hardware efficiency and functional capabilities. Current and planned efforts focus on memory optimization, training methodologies, and architectural expansions to address the inherent bottlenecks associated with high-frequency SRAM access.

Future iterations will focus on enhancing memory utilization through improved data packing. Currently, the implementation uses 49 BRAM tiles, but neuronal parameters and synaptic currents could be packed more efficiently

into single 48-bit words. This optimization aims to reduce the number of **memory access cycles** required for each neuron update and significantly decrease the overall on-chip **BRAM footprint**, allowing for even larger network topologies within the same Field Programmable Gate Array (FPGA) board.

On the algorithmic side, the integration of **Quantization-Aware Training (QAT)** is being explored. By incorporating quantization effects directly into the gradient descent process via snnTorch, the network can better adapt to lower precision formats. Preliminary investigations suggest that Quantization Aware Training (QAT) could maintain, or even improve, classification accuracy when utilizing weights quantized to fewer than 8 bits, despite the increased complexity of the gradient landscape during optimization.

Furthermore, a significant direction involves the implementation of a **convolutional front-end** to filter input event streams. A primary limitation of the current **fully connected architecture** is the large dimension of the input layer, which accounts for the majority of the memory consumption. Utilizing a **spiking convolutional layer** as a feature extractor would significantly reduce the input dimensionality, allowing the adLIF network to function as a dense backbone within a more complex system while optimizing hardware resources.

Finally, to mitigate the latency drawbacks observed in larger topologies, future work will also explore the introduction of **architectural parallelism**. Currently, the duration of each timestep depends on the number of neurons, with a predictable correlation ($R^2 > 0.99$) with the number of input events. Including parallel Processing Elements (PEs) would reduce the slope of the number of cycles per input spike (γ). This architectural shift would allow for a significant increase in **network size** or an enhancement of the **temporal resolution** (Δt) for the same topology without violating real-time feasibility constraints.

Appendix A

Memory Map and Register Interface

The communication between the external host and the accelerator core is managed via an AHB-Lite interconnect. The system memory space is partitioned into five segments, each dedicated to a specific functional block. This mapping allows the host to configure the network, provide input data, and monitor the internal states through a unified memory-mapped interface. The high-level memory distribution is summarized in Table A.1.

Table A.1: System Global Memory Map. Distribution of the 27-bit address space among the AHB slaves.

Target Module	**Address Range**	**Function**
Register File	`0x0_0000` - `0x0_0012`	Main configuration and core control interface.
SRAM	`0x1_0000` - `0x1_FFFF`	Storage for synaptic weights and neuronal parameters.
Output FIFO	`0x2_0000` - `0x2_0000`	Direct access to the Spike Output Stream.
Input FIFO	`0x3_0000` - `0x3_0000`	Direct access to the Event Input Stream.
Debug Registers	`0x4_0000` - `0x4_0026`	Internal signals and FSM states monitoring.

A.1 Register File

The Register File provides the primary interface to control and monitor the accelerator state. It includes registers for setting memory base addresses, defining layer topologies, and triggering the inference execution.

Table A.2: Register File Memory Map

Offset	Register Name	Access	Description
0x00	IN_SPK_NUM	R/W	Total number of input spikes to process.
0x01	IN_SPK_ADDR	R/W	Memory base address for input spike data.
0x02	START_TICK_GEN	R/W	Command to initiate the Tick Generator.
0x03	TICK_COUNTER	R/W	Configurable period for the Tick Generator.
0x04	L1_ADDR	R/W	Base address for Layer 1 parameters/weights.
0x05	L2_ADDR	R/W	Base address for Layer 2 parameters/weights.
0x06	L1_NRN_NUM	R/W	Total neuron count in Layer 1.
0x07	L2_NRN_NUM	R/W	Total neuron count in Layer 2.
0x08	START_SNN	W	Trigger to start SNN inference.
0x09	OUT_FIFO_FULL	R	Status: Output Spike FIFO full.
0x0A	OUT_FIFO_EMPTY	R	Status: Output Spike FIFO empty.
0x0B	SPIKE_RDY	R	Status: adLIF core has a spike ready.
0x0C	END_OF_TS	R	Status: Processing of current timestep completed.
0x0D	RUNNING	R	Status: Accelerator currently active.
0x0E	DEB_REG	R/W	Debug register for read/write verification.
0x0F	RSTN	W	Software active-low reset.
0x10	BS_VERSION	R	Returns the current bitstream version.
0x11	IN_FIFO_FULL	R	Status: Input Spike FIFO full.
0x12	IN_FIFO_EMPTY	R	Status: Input Spike FIFO empty.

A.2 Debug Registers

When enabled, the Debug Register block offers visibility into the processing core. It is particularly used for hardware-in-the-loop verification, exposing the intermediate states of the ADLIF pipeline.

Table A.3: Debug Register File Memory Map and Intermediate Pipeline

Offset	Register Name	Access	Description / Operation
FSM, Performance and Context Registers			
0x00	STATE_ADLIF	R	Top-level control FSM state.
0x01	STATE_INSPK	R	Input spike loading FSM state.
0x02	STATE_LAYER_CONTROLLER	R	Layer Controller FSM state.
0x03	STATE_NRN_LOAD	R	Neuron parameter loading FSM state.
0x04	STATE_NRN_UPDATE	R	Neuron dynamics calculation FSM state.
0x05	STATE_NRN_UPDATE_WRITE	R	Neuron state write-back FSM state.
0x07	STATE_PS	R	Post-synaptic processing FSM state.
0x08	STATE_WLOAD	R	Weight loading FSM state.
0x09	TOT_CC	R	Total clock cycles for current sample.
0x0A	CURR_TS_CC	R	Clock cycles consumed for current timestep.
0x0B	WSTATES_CC	R	Total wait states (SRAM stalls) encountered.
0x0C	CURR_LAYER	R	Active network layer index.
0x0D	CURR_NRN	R	Active neuron index within the layer.
0x0E	CURR_TS	R	Current simulation timestep counter.
Neuron Calculation Intermediate Pipeline			
0x0F	ADAPTED_I	R	$I - w$ (Input minus adaptation).
0x10	ONE_ALPHA	R	$1 - \alpha$ (Complement of potential decay).
0x11	ONE_BETA	R	$1 - \beta$ (Complement of adaptation decay).
0x12	SCALED_I	R	Scaled input current I_{scaled}.
0x13	ALPHA_U	R	$\alpha \cdot u$ (Decayed membrane potential).
0x14	U_BAR	R	Membrane potential candidate $\hat{u}$.
0x15	BETA_W	R	$\beta \cdot w$ (Decayed adaptation current).
0x16	A_UBAR	R	$a \cdot \hat{u}$ (Adaptation coupling).

Table A.3 – continued

Offset	Register Name	Access	Description / Operation
0x17	NEW_U	R	Final updated u after spike check.
0x18	BETA_SCALED_U	R	$\beta \cdot$scaled_u intermediate value.
0x19	NEW_W	R	Final updated adaptation current w.
0x1A	AU_P_B	R	$a \cdot u + b$ (Sub-threshold adaptation).
Neuron Parameters and Control			
0x1B	END_M_UPDATE	R	Status: memory update phase completed.
0x1C	NXT_SPIKE_ADDR	R	Memory address for next spike integration.
0x1F	N_PAR_I	R	Current synaptic input current (I).
0x20	N_PAR_W	R	Current adaptation current (w).
0x21	N_PAR_ALPHA	R	Decay parameter α.
0x22	N_PAR_U	R	Current membrane potential (u).
0x23	N_PAR_TH	R	Firing threshold ϑ.
0x24	N_PAR_BETA	R	Decay parameter β.
0x25	N_PAR_A	R	Sub-threshold coupling a.
0x26	N_PAR_B	R	Spike-triggered adaptation b.

Bibliography

[1] Christoph Posch et al. "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output". In: *Proceedings of the IEEE* 102.10 (2014), pp. 1470–1484. DOI: 10.1109/JPROC.2014.2346153.

[2] Aya Zuhair Salim and Luma Issa Abdul-Kareem. "A Review of Advances in Bio-Inspired Visual Models Using Event-and Frame-Based Sensors". In: *Advances in Technology Innovation* 10.1 (Jan. 2025), pp. 44–57. DOI: 10.46604/aiti.2024.14121.

[3] Md Abdullah-Al Kaiser et al. *Energy-Efficient & Real-Time Computer Vision with Intelligent Skipping via Reconfigurable CMOS Image Sensors*. 2024. arXiv: 2409.17341 [cs.CV].

[4] Yi Luo and Shahriar Mirabbasi. "A 60-Frames/s CMOS Image Sensor With Pixelwise Conversion Gain Modulation and Self-Triggered ADCs for Per-Frame Adaptive DCG-HDR Imaging". In: *IEEE Journal of Solid-State Circuits* 60.2 (2025), pp. 568–578. DOI: 10.1109/JSSC.2024.3433003.

[5] Raúl Taranco, José-María Arnau, and Antonio González. "δLTALTA: Decoupling Camera Sampling from Processing to Avoid Redundant Computations in the Vision Pipeline". In: *2023 56th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2023, pp. 1029–1043. DOI: 10.1109/MICRO59677.2023.00075.

[6] Xiangxiang Dai et al. "RESPIRE: Reducing Spatial–Temporal Redundancy for Efficient Edge-Based Industrial Video Analytics". In: *IEEE Transactions on Industrial Informatics* 18.12 (2022), pp. 9324–9334. DOI: 10.1109/TII.2022.3162598.

[7] Daoyu Li, Liheng Bian, and Jun Zhang. "High-Speed Large-Scale Imaging Using Frame Decomposition From Intrinsic Multiplexing of Motion". In: *IEEE Journal of Selected Topics in Signal Processing* 16.4 (2022), pp. 700–712. DOI: 10.1109/JSTSP.2022.3164524.

[8] Vivek Mohan et al. “EBBINNOT: A Hardware-Efficient Hybrid Event-Frame Tracker for Stationary Dynamic Vision Sensors”. In: *IEEE Internet of Things Journal* 9.21 (2022), pp. 20902–20917. DOI: 10.1109/JIOT.2022.3178120.

[9] N. Faramarzpour, M.J. Deen, and S. Shirani. “An Approach to Improve the Signal-to-Noise Ratio of Active Pixel Sensor for Low-Light-Level Applications”. In: *IEEE Transactions on Electron Devices* 53.9 (2006), pp. 2384–2391. DOI: 10.1109/TED.2006.881053.

[10] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. “A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor”. In: *IEEE Journal of Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[11] Alejandro Linares-Barranco et al. “Low Latency Event-Based Filtering and Feature Extraction for Dynamic Vision Sensors in Real-Time FPGA Applications”. In: *IEEE Access* 7 (2019), pp. 134926–134942. DOI: 10.1109/ACCESS.2019.2941282.

[12] Christian Brandli et al. “A 240 × 180 130 dB 3 μs Latency Global Shutter Spatiotemporal Vision Sensor”. In: *IEEE Journal of Solid-State Circuits* 49.10 (2014), pp. 2333–2341. DOI: 10.1109/JSSC.2014.2342715.

[13] K.A. Boahen. “Point-to-point connectivity between neuromorphic chips using address events”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47.5 (2000), pp. 416–434. DOI: 10.1109/82.842110.

[14] Maxence Bouvier et al. “Spiking Neural Networks Hardware Implementations and Challenges: a Survey”. In: *CoRR* abs/2005.01467 (2020). arXiv: 2005.01467.

[15] Patrick Plagwitz et al. *To Spike or Not to Spike? A Quantitative Comparison of SNN and CNN FPGA Implementations*. 2023. arXiv: 2306.12742 [cs.AR].

[16] Yiting Dong, Dongcheng Zhao, and Yi Zeng. “Temporal Knowledge Sharing Enable Spiking Neural Network Learning From Past and Future”. In: *IEEE Transactions on Artificial Intelligence* 5.7 (2024), pp. 3524–3534. DOI: 10.1109/TAI.2024.3374268.

[17] Evangelos Stromatias et al. “An Event-Driven Classifier for Spiking Neural Networks Fed with Synthetic or Dynamic Vision Sensor Data”. English. In: *Frontiers in Neuroscience* 11 (June 2017). DOI: 10.3389/fnins.2017.00350.

[18] Michael Pfeiffer and Thomas Pfeil. “Deep Learning With Spiking Neurons: Opportunities and Challenges”. English. In: *Frontiers in Neuroscience* 12 (Oct. 2018). DOI: 10.3389/fnins.2018.00774.

[19] Maximilian Baronig et al. “Advancing spatio-temporal processing through adaptation in spiking neural networks”. en. In: *Nature Communications* 16.1 (July 2025), p. 5776. DOI: 10.1038/s41467-025-60878-z.

[20] Maryam Zare, Elnaz Zafarkhah, and Nima S. Anzabi-Nezhad. “An area and energy efficient LIF neuron model with spike frequency adaptation mechanism”. In: *Neurocomputing* 465 (Nov. 2021), pp. 350–358. DOI: 10.1016/j.neucom.2021.09.004.

[21] Yunhua Chen et al. “An adaptive threshold mechanism for accurate and efficient deep spiking convolutional neural networks”. In: *Neurocomputing* 469 (Jan. 2022), pp. 189–197. DOI: 10.1016/j.neucom.2021.10.080.

[22] Lucas Deckers et al. “Co-learning synaptic delays, weights and adaptation in spiking neural networks”. English. In: *Frontiers in Neuroscience* 18 (Apr. 2024). DOI: 10.3389/fnins.2024.1360300.

[23] Eugene M. Izhikevich. “Resonate-and-fire neurons”. In: *Neural Networks* 14.6 (July 2001), pp. 883–894. DOI: 10.1016/S0893-6080(01)00078-8.

[24] Darjan Salaj et al. “Spike frequency adaptation supports network computations on temporally dispersed information”. In: *eLife* 10 (July 2021). Ed. by Timothy O’Leary, Timothy E Behrens, and Gabrielle Gutierrez, e65459. DOI: 10.7554/eLife.65459.

[25] Youngeun Kim and Priyadarshini Panda. “Revisiting Batch Normalization for Training Low-Latency Deep Spiking Neural Networks From Scratch”. English. In: *Frontiers in Neuroscience* 15 (Dec. 2021). DOI: 10.3389/fnins.2021.773954.

[26] Yufei Guo et al. “Membrane Potential Batch Normalization for Spiking Neural Networks”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 19363–19373. DOI: 10.1109/ICCV51070.2023.01779.

[27] Chang-Song Deng and Wei Liu. “Semi-implicit Euler–Maruyama method for non-linear time-changed stochastic differential equations”. en. In: *BIT Numerical Mathematics* 60.4 (Dec. 2020), pp. 1133–1151. DOI: 10.1007/s10543-020-00810-7.

[28] Arm Limited. *AMBA AHB Protocol Specification*. en. Issue C. Document ID: ARM IHI 0033C. Arm Limited. 2021.

[29] Jiajun Wu et al. “Efficient Design of Spiking Neural Network With STDP Learning Based on Fast CORDIC”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.6 (2021), pp. 2522–2534. DOI: 10.1109/TCSI.2021.3061766.

[30] Weilun Huang and Guolun Sheng. “Analysis and Research on UART Communication Protocol”. In: *2024 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*. 2024, pp. 768–771. DOI: 10.1109/ACCTCS61748.2024.00140.

[31] Dongcheng Zhao et al. *Improving Stability and Performance of Spiking Neural Networks through Enhancing Temporal Consistency*. 2023. arXiv: 2305.14174 [cs.NE].

[32] Guang Chen et al. “Event-Based Neuromorphic Vision for Autonomous Driving: A Paradigm Shift for Bio-Inspired Visual Sensing and Perception”. In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 34–49. DOI: 10.1109/MSP.2020.2985815.

[33] Davide Zoni and Andrea Galimberti. “Cost-effective fixed-point hardware support for RISC-V embedded systems”. In: *J. Syst. Archit.* 126 (2022), p. 102476. DOI: 10.1016/j.sysarc.2022.102476.

[34] “7 Series DSP48E1 Slice User Guide (UG479)”. en. In: (2018).

[35] Dexter Le et al. “A Review of Memory Wall for Neuromorphic Computing”. In: *2025 IEEE 4th International Conference on Computing and Machine Intelligence (ICMI)*. 2025, pp. 1–6. DOI: 10.1109/ICMI65310.2025.11141074.

[36] Manon Dampfhoffer et al. “Are SNNs Really More Energy-Efficient Than ANNs? an In-Depth Hardware-Aware Study”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 7.3 (2023), pp. 731–741. DOI: 10.1109/TETCI.2022.3214509.

[37] Zhuoer Li et al. “Efficiency analysis of artificial vs. Spiking Neural Networks on FPGAs”. In: *Journal of Systems Architecture* 133 (Dec. 2022), p. 102765. DOI: 10.1016/j.sysarc.2022.102765.

[38] Nitin Rathi et al. “Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware”. In: *ACM Comput. Surv.* 55.12 (Mar. 2023). DOI: 10.1145/3571155.

[39] Mattia Tambaro et al. “Influence of Reduced Numerical Precision in Spiking Neural Network Hardware Implementation”. In: *2025 International Conference on IC Design and Technology (ICICDT)*. 2025, pp. 85–88. DOI: 10.1109/ICICDT65192.2025.11078101.

[40] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. "Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details". English. In: *Frontiers in Neuroscience* 9 (Dec. 2015). DOI: 10.3389/fnins.2015.00481.

[41] Garrick Orchard et al. "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades". English. In: *Frontiers in Neuroscience* 9 (Nov. 2015). DOI: 10.3389/fnins.2015.00437.

[42] Gregor Lenz et al. *Tonic: event-based datasets and transformations.* 2021.

[43] Li Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142. DOI: 10.1109/MSP.2012.2211477.

[44] Jason K Eshraghian et al. "Training spiking neural networks using lessons from deep learning". In: *Proceedings of the IEEE* 111.9 (2023), pp. 1016–1054.

[45] Jason K. Eshraghian et al. "Training Spiking Neural Networks Using Lessons From Deep Learning". In: *Proceedings of the IEEE* 111.9 (2023), pp. 1016–1054. DOI: 10.1109/JPROC.2023.3308088.

[46] Friedemann Zenke and Surya Ganguli. "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks". In: *Neural Computation* 30.6 (June 2018), pp. 1514–1541. DOI: 10.1162/neco_a_01086. eprint: https://direct.mit.edu/neco/article-pdf/30/6/1514/1039264/neco_a_01086.pdf.

[47] Haibing Wang et al. "TripleBrain: A Compact Neuromorphic Hardware Core With Fast On-Chip Self-Organizing and Reinforcement Spike-Timing Dependent Plasticity". In: *IEEE Transactions on Biomedical Circuits and Systems* 16.4 (2022), pp. 636–650. DOI: 10.1109/TBCAS.2022.3189240.

[48] Michael Losh and Daniel Llamocca. "A Low-Power Spike-Like Neural Network Design". In: *Electronics* 8.12 (2019). DOI: 10.3390/electronics8121479.

[49] Daniel Valencia and Amir Alimohammad. "A generalized hardware architecture for real-time spiking neural networks". en. In: *Neural Computing and Applications* 35.24 (Aug. 2023), pp. 17821–17835. DOI: 10.1007/s00521-023-08650-6.

[50] Edris Zaman Farsa, Arash Ahmadi, and Oliver Keszocze. “Reconfigurable Digital FPGA Implementations for Neuromorphic Computing: A Survey on Recent Advances and Future Directions”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 9.5 (2025), pp. 3210–3232. DOI: 10.1109/TETCI.2025.3551934.

[51] Jin Zhang and Lei Zhang. “Spiking Neural Network Implementation on FPGA for Multiclass Classification”. In: *2023 IEEE International Systems Conference (SysCon)*. 2023, pp. 1–8. DOI: 10.1109/SysCon53073.2023.10131076.

[52] Yang Dan and Mu-ming Poo. “Spike Timing-Dependent Plasticity of Neural Circuits”. In: *Neuron* 44.1 (Sept. 2004), pp. 23–30. DOI: 10.1016/j.neuron.2004.09.007.

[53] Nazeerah Abdul Rahman and Nooraini Yusoff. “Modulated spike-time dependent plasticity (STDP)-based learning for spiking neural network (SNN): A review”. In: *Neurocomputing* 618 (Feb. 2025), p. 129170. DOI: 10.1016/j.neucom.2024.129170.

Acknowledgements

First and foremost, I would like to thank Thales Alenia Space for supporting and making this research activity possible. In particular, a special thanks goes to Gianluca Aranci, who has followed the project over these three years. I am truly grateful for his kindness and availability, especially for his patience whenever my communication regarding project updates wasn't exactly timely.

I would also like to thank Luca Giganti. During our time working together, he always found the time to satisfy my curiosity, helping me navigate the complex dynamics and terminology of the space sector. Part of his work methodology significantly guided me in the development of this project. My gratitude also goes to all the other colleagues at Thales, who were always friendly and ready to share a joke even in those moments when I felt a bit adrift.

I wish to express my gratitude to Prof. Marcello De Matteis for supervising my thesis work and enabling this research path.

A huge thank you goes to Mattia Tambaro and Lorenzo Stevenazzi: the idea for this network was born together, and working with you has been fantastic. Our exchange has been the greatest stimulus throughout this journey.

I am deeply grateful to Gianluca Furano for his warm welcome and for the opportunity to spend time at ESA. It was an incredible experience, made possible by people like him who know how to build the right connections and bring out the best in those around them.

While I cannot thank everyone I met over these years individually, I want to acknowledge all my colleagues at Bicocca, even those who were only there for a short time. You were the fuel that kept me going.

Finally, my most important thanks go to my family and my lifelong friends, who have been by my side since long before this experience began and who have always been my safe harbor.