Horizon Europe Framework Programme

Towards an AI-native, user-centric air interface for 6G networks

---

D2.4 – Mixed Analog-Digital Hardware Architecture

Contractual Delivery Date:	30-06-2025
Actual Delivery Date:	24-06-2025
Editor: *(name, organization)*	Zihang Song, Bipin Rajendran and Osvaldo Simeone, King's College London
Deliverable nature:	R
Dissemination level:	PU
Version:	1.0
Keywords: Neuromorphic computing, Hybrid analog-digital architecture, In-memory computing, Spiking neural networks, Energy-efficient hardware, AI-native transceivers	

ABSTRACT

One of the main objectives of WP2 is to develop energy-efficient hardware platforms for AI-native transceivers, with a focus on enabling low-power, high-performance signal processing at the wireless edge. This deliverable presents the outcomes of hardware architecture investigations conducted in Tasks

This project has received funding from the European Union's Smart Networks and Services Joint Undertaking (SNS JU) under grant agreement No. 101096379.

T2.1.2, T2.1.3, T2.2.3, and T2.2.4, centered on a hybrid analog-digital neuromorphic computing paradigm.

To address the energy bottlenecks in neural inference, particularly for sequence modeling tasks such as adaptive symbol detection, we propose a mixed-signal architecture that combines analog in-memory computing for feed-forward layers with digital stochastic circuits for attention mechanisms. This architecture supports real-time processing of temporally encoded spike signals and reduces memory access overhead by co-locating computation and storage. In addition, hardware-aware training and drift compensation techniques are implemented to mitigate the impact of device nonidealities.

Our design is evaluated on a spiking neural receiver use case. Experiments demonstrate significant improvements in computational and energy efficiency compared to conventional digital implementations, achieving up to 14.5× energy reduction and over 7× speed-up, while maintaining comparable detection accuracy. These results validate the feasibility of neuromorphic mixed-signal hardware for future 6G transceiver platforms and highlight the importance of hardware-software co-design in energy-constrained AI systems.

Disclaimer

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101096379. This publication reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Impressum

Full project title: Towards an AI-native, user-centric air interface for 6G networks

Short project title: CENTRIC

Number and title of the work package: WP2: AI-AI Hardware Platforms and Enablers

Number and title of task: T2.1: AI Computing Platforms, T2.2: Optimization for Real-Time AI Computing

Document title: Mixed Analog-Digital Hardware Architecture

Editor: Osvaldo Simeone, King's College London

Work-package leader: Osvaldo Simeone, Bipin Rajendran, King's College London

---

Executive summary

This document reports the outcomes of the mixed analog-digital hardware architecture research conducted in WP2 of the CENTRIC project. The objective is to enable energy-efficient neuromorphic inference for AI-native transceivers in future 6G networks. The work combines emerging memory technologies with neuromorphic computing principles to support adaptive, low-power signal processing.

The deliverable brings together the results of four interrelated subtasks:

- **T2.1.2 – Mixed Analog-Digital In-Memory Architecture:** This task established a high-level architectural framework for in-memory computing using memristive devices. The design incorporates crossbar-based cores to accelerate multiply-accumulate operations, along with supporting peripheral circuits, analog conductance sensing, and ADCs to ensure computation precision and robustness.
- **T2.1.3 – Neuromorphic Computing Architecture:** Building upon the T2.1.2 architecture, this subtask reconfigured the system to support spike-based computation. Event-driven readout circuits and inter-core spike communication were developed to implement spiking neuron models such as leaky integrate-and-fire and probabilistic spiking, using sub-threshold CMOS and stochastic PCM behavior.
- **T2.2.3 – Optimization for In-Memory Computing:** This subtask focused on mapping software-trained neural networks onto the analog hardware. It evaluated the impact of nonidealities such as programming stochasticity, conductance drift, and read noise, and proposed design refinements to improve throughput, energy efficiency, and accuracy under realistic operating conditions.
- **T2.2.4 – Optimization for Neuromorphic In-Memory Computing:** This task assessed the deployment of spiking neural networks on the neuromorphic hardware developed in T2.1.3. Performance was benchmarked in terms of latency, energy consumption, and throughput, with comparisons to general-purpose platform via emulation environments.

Together, these efforts resulted in a mixed-signal neuromorphic hardware platform that fuses analog in-memory computing with digital spiking logic. The implemented demonstrator supports in-context learning for adaptive symbol detection and achieves significant energy efficiency gains, which demonstrates potential as a foundational architecture for next-generation 6G transceivers.

List of authors

Company	Author	Contribution
KCL	Osvaldo Simeone	Editor
KCL	Bipin Rajendran	Editor
KCL	Zihang Song	Contributor
KCL	Prabodh Katti	Contributor

Table of Contents

List of Figures

List of Tables

Abbreviations

AC	Accumulate
ADC	Analog-to-Digital Converter
AIMC	Analog In-Memory Computing
ALU	Arithmetic Logic Unit
ANN	Artificial Neural Network
BER	Bit Error Rate
BL	Bit Line
BPTT	Backpropagation Through Time
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
CSA	Carry-Save Adder
CT	Conventional Training
DAC	Digital-to-Analog Converter
DIMC	Digital In-Memory Computing
FIFO	First-In First-Out
FP	Floating Point
GDC	Global Drift Compensation
GPU	Graphics Processing Unit
HWAT	Hardware-Aware Training
ICL	In-Context Learning
LFSR	Linear Feedback Shift Register
INT	Integer
LIF	Leaky Integrate-and-Fire
MAC	Multiply-Accumulate
MIMO	Multiple Input Multiple Output
MMSE	Minimum Mean Squared Error
MUX	Multiplexer
MVM	Matrix-Vector Multiplication
NC	No Compensation
NVM	Non-Volatile Memory
PCM	Phase Change Memory
QPSK	Quadrature Phase Shift Keying
ReRAM	Resistive Random Access Memory
SA	Synaptic Array
SAC	Stochastic Attention Cell
SAR	Successive Approximation-Register
SNN	Spiking Neural Network
SNR	Signal-to-Noise Ratio
SRAM	Static Random Access Memory
SSA	Stochastic Spiking Attention
WL	Word Line

1 Introduction

The CENTRIC project is dedicated to developing AI-native and energy-efficient technologies tailored for 6G wireless systems. A critical element in this vision is the innovation of transceiver architectures that can perform fast, adaptive, and low-power signal processing directly at the wireless edge. Conventional digital transceiver designs struggle to meet these stringent demands due to their inherently high energy consumption and rigid computational frameworks. Neuromorphic computing, inspired by the brain's event-driven and parallel processing characteristics, emerges as a promising alternative. It is ideally suited to address the sparse, intermittent, and bursty nature of wireless communication signals.

This deliverable thoroughly explores how neuromorphic computing principles can be effectively adapted and integrated into wireless transceiver hardware. Specifically, we focus on the implementation of key neuromorphic functionalities—spiking neural networks (SNNs)—using hybrid analog-digital hardware platforms. We leverage analog in-memory computing (AIMC) to handle feedforward neural operations efficiently and implement stochastic attention mechanisms for efficient sequential processing. Furthermore, we develop specialized training and calibration methodologies to manage and mitigate imperfections inherent in analog hardware. These methods ensure precise, robust, and reliable inference for complex symbol detection tasks.

1.1 Neuromorphic Computing for Adaptive Neural Receivers

The increasing deployment of wireless receivers at the network edge introduces severe constraints on energy availability, latency, and adaptability. Traditional digital signal processing pipelines often operate in fixed stages—training, channel estimation, and symbol detection—each consuming power and introducing latency. In contrast, neuromorphic computing offers a biologically inspired alternative that processes signals in an event-driven and energy-efficient fashion using SNNs [1].

In this deliverable, we extend the neuromorphic design to support in-context learning (ICL), where a receiver learns to infer new symbols by recognizing patterns within pilot-reference sequences, without needing an explicit training or channel estimation phase [2] [3]. This formulation makes the receiver highly adaptive to dynamically varying channel conditions, enabling fast response to new users or interference scenarios with minimal pilot overhead.

Aligned with Subtask T2.1.3, we implement a spiking neural receiver that encodes wireless signals as temporal spike sequences and processes them through a Transformer-based architecture optimized for ICL [2]. The model exploits sparsity and sequence structure, enabling the system to learn channel behavior implicitly from a few examples. Combined with neuromorphic execution, this approach lays the groundwork for ultra-low-power, latency-efficient 6G receiver hardware.

1.2 Hybrid Analog-Digital Hardware Acceleration for Spiking Transformers

Transformer architectures offer powerful capabilities for sequence modeling and ICL. However, their reliance on dense matrix multiplications and nonlinear attention

computations makes them inherently energy-intensive when deployed on conventional digital hardware. AIMC addresses part of this challenge by executing matrix-vector multiplications (MVMs) directly within memory arrays, thereby reducing data movement overhead and associated power consumption [4].

However, AIMC is not well suited for implementing attention mechanisms due to their dynamic and nonlinear nature. To overcome the limitation, we propose a hybrid spiking Transformer architecture in which AIMC cores handle the feedforward path, and a complementary digital subsystem performs attention using spike-based stochastic logic.

We design phase-change memory (PCM)-based AIMC cores to support multi-bit MVM operations with reduced energy overhead, as addressed in Task T2.1.2. To overcome the inefficiency of AIMC in nonlinear attention, we develop a spike-based stochastic attention mechanism that operates on Bernoulli-encoded inputs using simple logic gates [5]. This design, aligned with the objectives of Tasks T2.1.3 and T2.2.4, avoids multipliers and softmax units entirely. In parallel, we introduce training and calibration procedures under Task T2.2.3 to ensure robustness against analog nonidealities such as noise, drift, and device mismatch. The resulting architecture supports event-driven inference with low energy consumption and hardware reliability across dynamic communication scenarios.

1.3 Spiking Neural Receiver: Training, Hardware Adaptation, and Demonstration

The integrated training, inference, and evaluation of the spiking neural receiver are presented. The system deploys the neuromorphic algorithmic design described in Section 1.1 on the hybrid hardware acceleration platform introduced in Section 1.2.

To ensure robust performance under analog nonidealities, we apply hardware-aware training (HWAT) strategies that incorporate quantization effects, analog noise, and conductance drift into the optimization process [6]. These techniques yield models that remain resilient to the imperfections of AIMC and stochastic logic-based attention. Complementary calibration procedures, such as global drift compensation, further enhance long-term inference stability. The system-level evaluation employs a suite of tools—Sionna for wireless signal modeling, *SpikingJelly* for SNN training, IBM *AIHWKit* for analog inference simulation, and *DNN+NeuroSim V1.4* (*NeuroSim*) for energy and area estimation [7] [8] [9]. The final hardware-simulated receiver achieves low-latency symbol detection with competitive accuracy and substantial energy efficiency, demonstrating the viability of spiking neuro receivers for deployment in edge communication systems.

2 Neuromorphic Computing for Adaptive Neural Receivers

(*This section integrates contributions from T2.1.3 and lays conceptual groundwork for T2.2.4.*)

The growing demand for intelligent edge devices in wireless networks—especially in the context of 6G—places strict requirements on latency, power consumption, and adaptability. These demands pose a significant challenge to conventional ANNs, which operate under the assumption of continuous, synchronous data flow and dense computation. Such architectures are ill-suited for the inherently sparse, intermittent, and burst-driven nature of wireless signals observed in real-world communication systems.

Neuromorphic computing, inspired by the information processing mechanisms of biological brains, offers a promising alternative [1]. By leveraging SNNs, which operate via discrete, event-driven spikes instead of continuous activations, neuromorphic systems can respond to salient input features while remaining inactive otherwise [10]. This design principle leads to highly efficient computation, both in terms of energy and latency, and is particularly well-aligned with the bursty traffic and sporadic inference needs at the wireless edge.

2.1 Spiking Neural Networks: Core Principles

In contrast to ANNs, which propagate continuous-valued activations through each layer, SNNs communicate using binary spike signals that occur in time. At the core of SNNs are spiking neurons, most commonly implemented as leaky integrate-and-fire (LIF) neurons. The dynamics of an LIF neuron can be described mathematically by the following equation:

$$\mathrm{V(t)} = \beta \mathrm{V(t-1)} + \mathrm{I(t)}, \ \text{if } V(\mathrm{t}) \geq \mathrm{V_{threshold}}, \ \text{then } V(t) = 1 \text{and reset } V(t) = 0$$

Here, $V(\mathrm{t})$ represents the neuron's membrane potential at time t, $\beta \in [0,1)$ is a decay factor (leak), $I(\mathrm{t})$ is the input signal at time t, $\mathrm{V_{threshold}}$ is the firing threshold, and $O(\mathrm{t})$ is the output spike at time t.

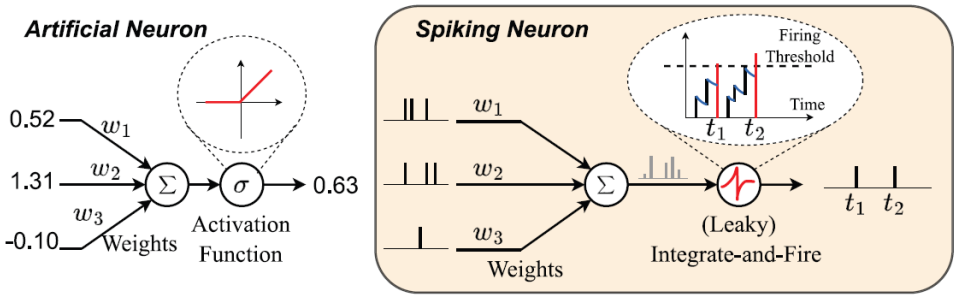

Figure 1 Comparison between artificial neuron and leaky integrate-and-fire spiking neuron.

2.2 Bernoulli Spike Encoding of Wireless Signals

To process real-valued wireless input signals with SNNs, the signals must be first converted into temporal spike sequences. A common method for doing this is Bernoulli spike encoding, which translates each scalar value into a binary sequence distributed over multiple time steps.

Given a normalized scalar input $x \in [0,1]$, the Bernoulli encoding generates a sequence $\{x_t\}_{t=1}^{T}$ such that:

$$x_t \sim Bern(x), t = 1,2, \dots, T$$

That is, each element $x_t \in \{0,1\}$ independently takes the value 1 with probability x, and 0 otherwise. Over time, the expected sum of the spike sequence approximates the original analog value, while maintaining a binary format suitable for SNN processing [11].

This encoding is particularly suited for wireless signals after quantization and normalization. Complex-valued baseband signals can be decomposed into their real and imaginary parts, each of which is normalized to the $[0,1]$ range before encoding. The result is a temporally sparse, spike-based representation that preserves signal fidelity while enabling event-driven processing.

2.3 In-Context Learning with Spiking Neural Networks for Wireless Signal Processing

Modern wireless environments are highly dynamic—characterized by unpredictable channel fading, interference, and user mobility. Traditional receivers address this variability through a three-phase pipeline: explicit channel estimation, followed by equalization, and finally symbol detection [12]. While effective, this pipeline introduces overhead from pilot transmission, inference latency, and the need for model re-training or re-estimation when channel conditions change.

An emerging alternative to this paradigm is ICL, a property initially observed in large transformer models [13] [14] [15]. ICL enables a model to *implicitly learn* a task by observing a small number of input-output examples—without gradient-based weight updates. When applied to wireless communications, this concept allows a neural receiver to adapt to channel variations *on the fly* by leveraging pilot symbols as contextual examples, without requiring explicit channel estimation or architectural reconfiguration [3].

2.3.1 ICL Formulation for Symbol Detection

In the context of symbol detection over a MIMO link, the task is to infer a transmitted symbol vector s from its noisy received version y, without access to the channel matrix H. The receiver is instead given a **context** of pilot pairs:

$$\mathcal{C} = \{(y_1, s_1), (y_2, s_2), \dots, (y_N, s_N)\}$$

Each pair in $\mathcal{C}$ is drawn from the same unknown channel and noise distribution as the test input y. The goal is to infer the corresponding symbol vector s purely by reasoning over the pilot examples and the new input:

$$(\mathcal{C}, y) \mapsto \hat{s}$$

This direct context-to-prediction mapping avoids channel inversion or statistical estimation and is naturally suited to neural architectures designed for sequence processing.

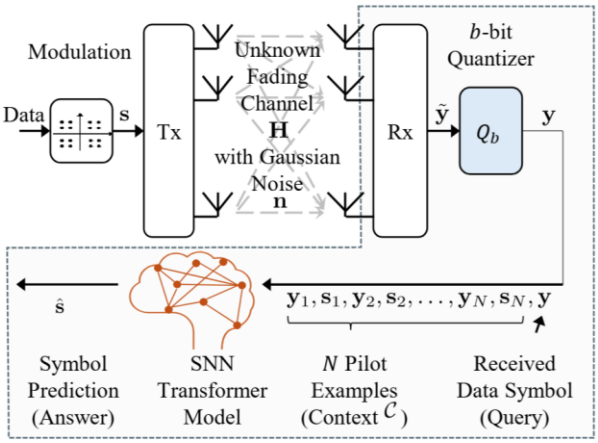

Figure 2 Illustration of in-context learning for MIMO symbol detection.

2.3.2 Implementing ICL with Spiking Transformers

Transformer-based architectures are particularly effective at ICL because of their self-attention mechanism, which allows the model to compare and align the query input with relevant context examples [16]. In an SNN framework, this mechanism must be adapted to operate with discrete-time spike sequences rather than continuous values [17].

Figure 2 shows how to apply ICL in a spiking setup, we first represent both the context set $\mathcal{C} = \{(y_i, s_i)\}_{i=1}^{N}$ and the query input y as spike sequences. The received signal vectors $y_i, y \in C^{N_r}$ are separated into their real and imaginary parts and normalized within the quantizer dynamic range $[l_{min}, l_{max}]$ucing real-valued vectors $y_i^{\mathrm{nr}}, y^{\mathrm{nr}} \in [0,1]^{2N_r}$. Similarly, the target symbol vectors $s_i \in \mathcal{S}^{N_t}$ (e.g., QPSK symbols) are converted into one-hot encodings and normalized to fall within the unit interval.

Each normalized input $x \in [0,1]^d$ is converted into a binary spike sequence $\{x^t\}_{t=1}^{T}$, where each spike vector $x^t \in \{0,1\}^d$ is generated via Bernoulli encoding as

$$x_j^t \sim \mathrm{Bern}(x_j), \quad \text{for } j = 1, \ldots, d,\ t = 1, \ldots, T.$$

This stochastic encoding preserves the expected value of the original input while enabling efficient logical computation over spikes.

At each time step t, this sequence is passed through a spiking token embedding layer implemented via a LIF neuron model with trainable embedding matrix $W_e \in R^{D_e \times d}$, producing

$$E_0^t = \mathrm{LIF}^t(W_e \mathcal{X}^t) \in \{0,1\}^{D_e \times (2N+1)}.$$

The embedded spike tokens are then processed by a stack of L spiking transformer decoder layers. Each layer contains:

- **A stochastic self-attention (SSA)** module that computes attention weights between tokens via spike-wise logic.
- **A feedforward LIF network** that applies temporal integration and nonlinearity.

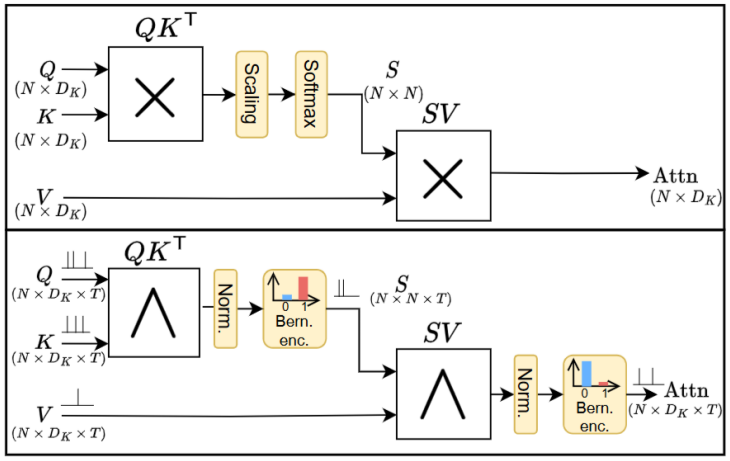

Figure 3 Top: A conventional implementation of an attention block based on real-valued multiply-and-accumulate operations within an ANN architecture. Bottom: The proposed SNN-based attention block with spiking inputs, outputs, and stochastic computations. Multiplication operations are replaced with logical AND (⋀) operations on spikes. Further hardware efficiency is achieved by the replacement of scaling and softmax blocks with a Bernoulli rate encoder.

The attention mechanism replaces standard dot-product attention with spike-wise AND operations and population counting. For each attention head h, queries, keys, and values are computed as:

$$Q_h^t = \mathrm{LIF}^t\left(W_Q E_{l-1}^t\right), \quad K_h^t = \mathrm{LIF}^t\left(W_K E_{l-1}^t\right), \quad V_h^t = \mathrm{LIF}^t\left(W_V E_{l-1}^t\right).$$

Different from conventional ANN-based attention mechanism which relies on heavy full-precision multiplications, we propose an efficient way to perform attention mechanism in spiking domain leveraging the binary nature of Q_h^t, K_h^t and V_h^t, as shown in Figure 3. The attention logits $\tilde{A}_{m,m'}^t$ between query token m and key token m' are computed by accumulating the logic AND (∧) operation results as

$$\tilde{A}_{m,m'}^t = \sum_{j=1}^{D_K} Q_{j,m}^t \wedge K_{j,m'}^t,$$

These logits are normalized and sampled as binary attention weights as

$$A_{m,m'}^t \sim \mathrm{Bern}\left(\frac{1}{D_K}\tilde{A}_{m,m'}^t\right).$$

The attention outputs are then computed using a second set of AND operations between the attention weights and the value tokens as

$$\tilde{F}_{j,m}^t = \sum_{m'=1}^{M} A_{m,m'}^t \wedge V_{j,m'}^t, \quad F_{j,m}^t \sim \mathrm{Bern}\left(\frac{1}{M}\tilde{F}_{j,m}^t\right).$$

The resulting values F^t are concatenated across heads and passed to a feedforward LIF block to complete the layer. This process is repeated across all layers, preserving spike-based temporal dynamics.

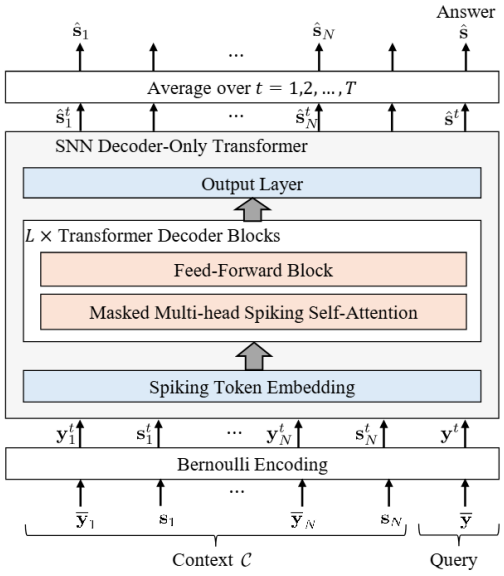

Figure 4: Block diagram of the proposed spiking transformer for ICL-based MIMO symbol detection.

Finally, at each time step, the final decoder output is projected via a linear matrix $W_O \in R^{KN_t \times D_e}$, producing logits o_{2N+1}^t for the query token. The output logits are temporally averaged over all T steps:

$$\hat{s} = \frac{1}{T}\sum_{t=1}^{T} W_O E_L^t[:, 2N+1],$$

and the final predicted symbol is obtained as the index of the maximum value in $\hat{s}$.

This results in an event-driven ICL system that adaptively maps pilot examples and query observations to symbol predictions with minimal computational overhead, as shown in Figure 4. Unlike conventional approaches that rely on weight updates, explicit channel estimation, or iterative optimization, this method performs inference entirely through fixed, pre-trained network parameters and feedforward computation over spike-encoded inputs. The temporal structure of the spike representation acts as an implicit memory mechanism, allowing the system to dynamically adapt to new channel conditions simply by observing a small number of context examples.

2.4 Training the Spiking ICL Network

To enable symbol detection via ICL, the spiking transformer is trained on a large set of synthetic wireless communication tasks, each characterized by a unique channel realization. The training objective is to learn a universal mapping that can generalize to unseen channels by leveraging pilot-context information, without requiring explicit channel estimation or weight updates at test time.

Each training task corresponds to a randomly sampled channel $H \in C^{N_r \times N_t}$ and noise variance σ^2. For each task, a set of pilot input–output pairs $\{(y_i, s_i)\}_{i=1}^N$ and a query input y_q with unknown symbol s_q are generated using the canonical quantized MIMO model:

$$y_i = Q_b(Hs_i + n_i), \quad n_i \sim \mathcal{CN}(0, \sigma^2 I),$$

with $s_i \in \mathcal{S}^{N_t}$ drawn from a modulation constellation (e.g., QPSK, 16-QAM) and $Q_b(\cdot)$ is a quantizer with quantization resolution b. Each complex vector is separated into real and imaginary parts and normalized to lie in $[0,1]$, then encoded into temporal binary sequences using Bernoulli sampling (see Section 2.2). These spike-encoded sequences are arranged into input tokens for the spiking transformer as described in Section 2.3.2.

The model is trained using episodic supervision: for each task, the model observes the spike-encoded pilot tokens and must correctly classify the query symbol vector. The output logits for the query are integrated over T time steps, and the average is passed through a softmax to produce a probability vector $\hat{p} \in [0,1]^{|\mathcal{S}| \times N_t}$. The loss function is the sum of cross-entropy losses over all transmit dimensions:

$$\mathcal{L} = \sum_{j=1}^{N_t} \mathrm{CE}(\hat{p}_j, s_{q,j}),$$

where $\hat{p}_j$ is the predicted symbol distribution for transmit stream j, and $s_{q,j}$ is the true one-hot encoded symbol.

Training is conducted using *PyTorch* and the *SpikingJelly* framework, which supports backpropagation through time (BPTT) for spiking networks using surrogate gradients. The channel sampling and pilot construction follow the Sionna simulator, ensuring realism in channel dynamics and symbol formation. The network is trained over thousands of independently drawn channel tasks, each treated as a distinct episode. Importantly, the model does not memorize or adapt to specific channels—it learns a task-agnostic strategy for inferring query outputs based solely on pilot examples.

The use of fixed-length binary spike sequences ensures that the temporal dynamics are preserved across tasks and that the model develops time-resolved attention mechanisms. Through this training paradigm, the network learns to align pilot patterns with query inputs and reason over symbol relationships in a biologically inspired, event-driven manner.

2.5 Evaluation of the Trained Spiking ICL Receiver

This section presents an experimental evaluation of the proposed SNN-based ICL receiver. The model is assessed in terms of its symbol detection accuracy and energy efficiency in a realistic 2×2 MIMO wireless scenario, using QPSK modulation and a wide range of channel conditions. The performance is benchmarked against two baselines: (1) an ideal MMSE equalizer with full channel knowledge, and (2) an ANN-based ICL transformer implementation following the architecture proposed in [Zecchin et al., 2023], configured to match the SNN model in terms of size and training.

2.5.1 Symbol Detection Performance

The SNN-based ICL model is trained over a set of randomly generated MIMO tasks, each defined by a unique channel realization $H \sim \mathcal{CN}(0,1)$ and noise variance σ^2. A fixed context set of $N = 20$ pilot symbols is used for adaptation, and performance is evaluated at an SNR of 10 dB. The model architecture comprises L = 4 decoder layers, an embedding dimension of

$D_e = 256$, and $n_h = 8$ attention heads. Inputs are quantized using a 4-bit mid-tread uniform quantizer with clipping boundaries $l_{\min} = -4$ and $l_{\max} = 4$.

Figure 5 (left) shows the bit error rate (BER) achieved by the SNN and ANN transformers as a function of the number of pre-training tasks N_{Train}, ranging from 2^0 to 2^{15}. The results reveal that the SNN achieves comparable performance to the ANN baseline when trained on a sufficient number of tasks. Notably, once $N_{\text{Train}} \geq 2^5$, the BER gap between the two models becomes negligible, and both models approach the accuracy of the ideal MMSE receiver. This demonstrates the strong generalization capacity of the spiking transformer, even in the presence of temporal spike-based encoding and logic-based attention.

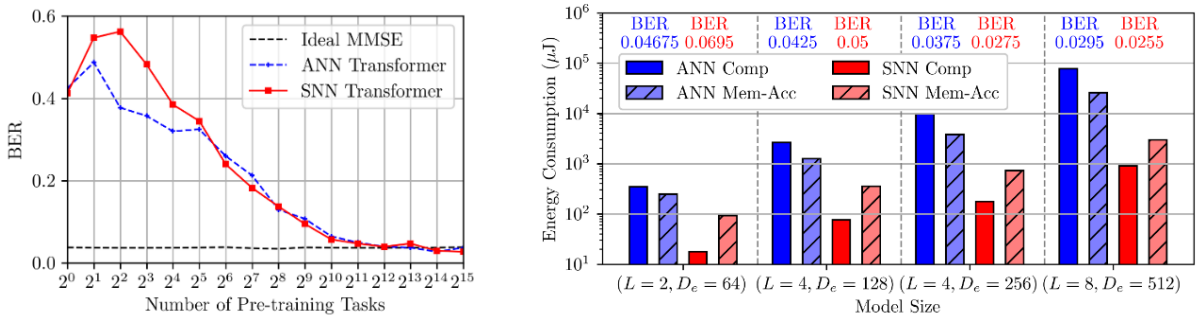

Figure 5 (Left) Detection BER for ANN and SNN transformers, pre-trained with varying numbers of tasks and tested under an SNR of 10 dB. Both models use 4 layers, 8 heads, and an embedding dimension of 256. (Right) Computing energy consumption and memory access energy consumption for ANN and SNN transformers of varying sizes, marked with the lowest BER achievable with a pre-training size of 32768. For the SNN model, the number of timesteps is set to T = 4 for both panels.

2.5.2 Efficiency Analysis

The SNN implementation offers significant improvements in energy efficiency compared to its ANN counterpart. The energy cost of inference is estimated by accounting for both computation and memory access, assuming 8-bit quantized parameters and activations, with data stored in on-chip SRAM. These estimates follow methodologies established in neuromorphic energy modeling literature [Pedram 2017; Buffa 2021; ACE-SNN 2022].

Figure 5 (right) presents the estimated energy consumption across several model configurations: (2,64), (4,128), (4,256), and (8,512). Each data point indicates the energy cost (compute + memory) and the corresponding BER at 10 dB SNR. For the smallest model configuration $(L = 2, D_e = 64)$, the SNN achieves a 20× reduction in computation energy and a 2.6× reduction in memory access energy compared to the ANN, at the cost of a slightly higher BER (0.070 vs. 0.047). As the model size increases, these energy advantages scale up further: the $(L = 8, D_e = 512)$ SNN model achieves an 86× reduction in computation energy and an 8.7× reduction in memory energy, while achieving better BER than the ANN (0.026 vs. 0.030).

These results demonstrate that spiking ICL architectures not only retain competitive accuracy but also unlock dramatic improvements in energy efficiency, particularly at larger model scales. The event-driven and sparse nature of spike-based computation is especially advantageous for latency-and energy-sensitive edge deployments, where minimizing memory movement and multiply-accumulate (MAC) operations is critical.

3 Hybrid Analog-Digital Hardware Acceleration for Spiking Transformers

(This section synthesizes outcomes from T2.1.2 and T2.2.3)

3.1 Motivation for Mixed-Signal Implementation

The integration of SNNs with transformer architectures has demonstrated strong potential in enabling efficient and adaptive sequence modeling for wireless signal processing tasks, such as symbol detection via ICL. As discussed in Section 2, spiking transformers leverage sparse, temporally-coded binary spikes to replace conventional dense activation propagation with lightweight accumulate (AC) operations. This leads to significant algorithmic advantages in terms of energy savings and event-driven adaptability [17].

However, despite their theoretical efficiency, spiking transformers remain computationally inefficient when deployed on conventional digital hardware platforms, such as GPUs or CPUs. These platforms are optimized for dense, high-precision workloads (e.g., FP32/FP16 arithmetic), whereas spiking models operate on binary or integer-valued, highly sparse temporal data. This mismatch manifests in multiple ways:

- **Poor resource utilization:** Spiking layers produce sparse binary outputs, but GPUs still allocate resources for full-precision MAC operations.
- **Temporal bottlenecks:** Spike-based processing unfolds over time steps, requiring repeated memory access and synchronization, resulting in high latency and energy overhead.
- **Memory inefficiency:** Intermediate results must be stored and retrieved frequently in digital memory, further compounding energy costs.

These limitations have motivated a growing interest in hardware-algorithm co-design for neuromorphic architectures. Yet, existing digital accelerators remain primarily tailored to ANNs, and even recent AIMC accelerators for transformers are built for ANN workloads—where inputs and weights are both continuous and dense.

By contrast, spiking transformers present distinct architectural opportunities and challenges. Their feedforward layers involve large-scale static weight matrices and sparse binary inputs—making them ideally suited for analog in-memory acceleration. However, attention layers compute dynamic weightings that must be recomputed per input query, making them unsuitable for static analog arrays and better suited for digital or logic-based solutions.

To address these complementary needs, we adopt a mixed-signal hardware approach: We use PCM-based AIMC for static-weight layers, where matrix weights are programmed once and reused efficiently without moving data. Furthermore, we implement SSA in lightweight digital logic, using spike-wise operations such as logical AND and counters to perform binary attention without multipliers or memory-intensive arithmetic.

This combination of AIMC and stochastic computing constitutes the first hardware-native accelerator design tailored to spiking transformers. In the next sections, we explain the

capabilities and limitations of AIMC (Section 3.2), detail the analog feedforward architecture (Section 3.3), and describe how we address this with logic-based SSA (Section 3.4).

3.2 AIMC for Spiking Transformers: Capabilities and Limitations

AIMC presents a promising direction for accelerating the feedforward components of spiking transformers by performing MVMs directly within memory arrays. This in-situ computation paradigm eliminates the need for extensive data movement between compute and memory units, which is a primary energy and latency bottleneck in conventional digital accelerators. Transformer architectures, where linear projections and feedforward layers dominate the computational load, can particularly benefit from such an approach.

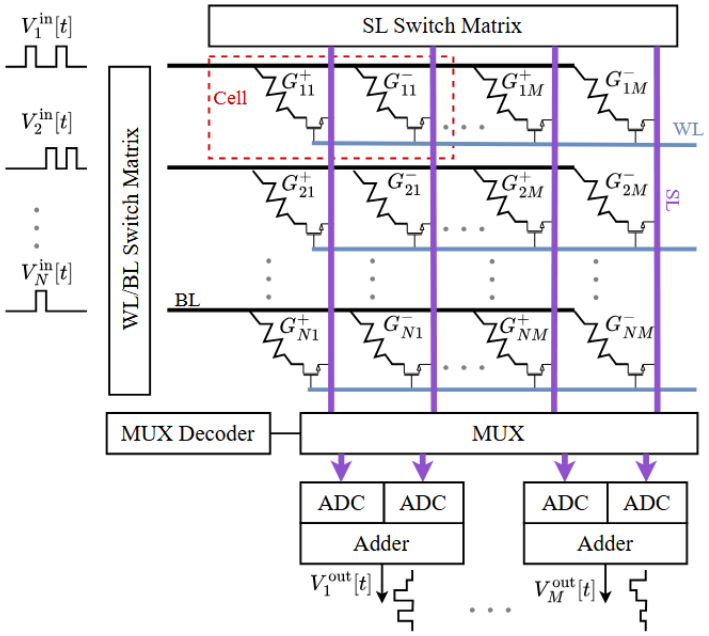

Figure 6 Illustration of a typical implementation of synaptic array with spike-encoded signals as input.

As shown in Figure 6,each AIMC unit consists of a synaptic array implemented as a crossbar of devices. These devices are arranged at the intersections of word lines (WLs) and bit lines (BLs), with their conductance levels programmed to represent matrix weights. To support signed weights, each matrix element is encoded using a differential pair of NVM devices. In our implementation, 4-bit quantization is employed to strike a balance between precision and analog programmability. During inference, the input vector is applied as voltages to the BLs. According to Ohm's Law, each active device produces a current proportional to its conductance and input voltage. The resulting currents from each column are summed at the source lines (SLs) based on Kirchhoff's Current Law, producing the analog dot product between input and stored weights.

These analog currents are sensed by multilevel current-mode amplifiers and converted into digital values via successive approximation-register (SAR) analog-to-digital converters (ADCs). To optimize area and latency, these readout circuits are time-multiplexed across columns using hardware multiplexers. In addition to the core compute path, peripheral components

such as buffers, decoders, and switching matrices are included to facilitate programmable control and efficient memory access.

In spiking transformers, the inputs to the feedforward layers are temporally distributed binary spike trains. This encoding allows the analog synaptic arrays to operate directly on one-bit signals, avoiding the need for digital-to-analog converters (DACs) during inference. DACs are only used in the initial phase to program conductance levels that represent trained weights. Once programmed, these weights remain static throughout inference, enabling constant-time $O(1)$ execution of MVMs and making AIMC ideally suited for embedding, projection, and fully connected layers.

However, despite its effectiveness in static-weight scenarios, **AIMC is not suitable for implementing attention mechanisms**. Attention layers rely on dynamic, input-dependent computation: each query token must be compared against keys, normalized, and used to reweight the values. This process requires intermediate matrices that change at every step, which AIMC cannot handle efficiently.

Using AIMC for attention would involve frequent writes to NVM arrays, incurring high latency, energy overhead, and accelerating device wear. Moreover, analog circuits struggle to reliably implement nonlinear functions such as softmax, further limiting their practicality. Prior work confirms that even ANN-based AIMC accelerators typically offload attention to digital processors due to these limitations.

Although spiking transformers simplify some of these computations using binary spike trains, attention weights still vary with every query. This prevents pre-programming in memory and renders static analog mapping ineffective. As a result, AIMC cannot efficiently realize attention in either spiking or conventional transformer architectures.

The next section introduces a hardware-native solution that overcomes this limitation through spike-compatible stochastic logic, enabling dynamic attention without analog memory writes or multipliers.

3.3 AIMC Engine: Spiking Neuron Tiles and Row-Block-Wise Mapping

The AIMC engine is responsible for executing the static-weight operations in the spiking transformer architecture, specifically the feedforward and fully connected layers. It is built from a tiled structure where each spiking neuron tile consists of synaptic arrays, LIF units, and local buffers. The goal of this architecture is to minimize data movement and enable local accumulation of analog computations with spike-based neuron dynamics.

3.3.1 Synaptic Array Architecture

Each synaptic array comprises a fixed-size PCM crossbar configured to support analog matrix-vector multiplication. Input spike-encoded vectors are streamed as voltage pulses across the word lines of the crossbar, and the resulting analog currents are collected along the bit lines. These currents represent the analog dot product between the input voltage and programmed conductance values.

To ensure reliability and mitigate parasitic effects, the crossbar size is constrained to 128×128 cells. This constraint necessitates the partitioning of large weight matrices across multiple synaptic arrays and possibly across multiple tiles. For example, a 384×512 weight matrix is divided into twelve 128×128 sub-blocks. Each synaptic array is equipped with 16 shared ADCs (due to the sharing ratio of 8), allowing digitization of 16 outputs per clock cycle. The ADC outputs are routed directly into local computation units to minimize global memory accesses. The detailed configuration parameters for the synaptic array is shown in Table 1.

Table 1 Synaptic Array Configuration Parameters

Parameter	Value
Resistive device	PCM
Conductance resolution	4 bits
Weight resolution	5 bits
Devices per cell	2
Crossbar dimension	128×128
ADC resolution	5 bits
ADC sharing ratio	8

3.3.2 Row-Block-Wise Mapping Strategy

The row-block-wise mapping strategy provides a systematic approach to assigning portions of a weight matrix to individual synaptic arrays and organizing computation within a tile. As shown in Figure 7, each row block of the full weight matrix is mapped to a set of synaptic arrays within a single tile. For example, in a 384×512 weight matrix, rows 129–256 are mapped to four 128×128 arrays (labeled SA 2-1 to SA 2-4) within Tile 2. The input vector of size 512×1 is segmented into four 128×1 subvectors and fed into these arrays in parallel.

Each synaptic array computes a local partial sum corresponding to its submatrix. These partial sums are not stored externally but are immediately routed to a shared LIF unit associated with each output feature. The LIF unit aggregates the local contributions using a carry-save adder, updates the membrane potential, and triggers a spike if the potential exceeds a predefined threshold. This process enables spike generation in a strictly local manner and avoids the overhead of storing intermediate analog or digital pre-activation values.

Each LIF unit integrates new pre-activations into its membrane state at every time step. If the membrane potential exceeds the stored threshold, a spike is emitted and the membrane state is reset. A digital right-shift operation is performed on the membrane register at each time step to emulate leaky integration, corresponding to a decay factor of \beta = 0.5. The comparator output is forwarded to the output buffer and subsequently to the shared SRAM.

To ensure alignment across the multiple synaptic arrays and their shared ADCs, a unified MUX decoding scheme is applied. During each MUX cycle, 16 output features are read simultaneously from each SA, and the same decoding configuration is applied to all SAs within a tile. This guarantees that the local sums are correctly aligned and accumulated within their corresponding LIF units.

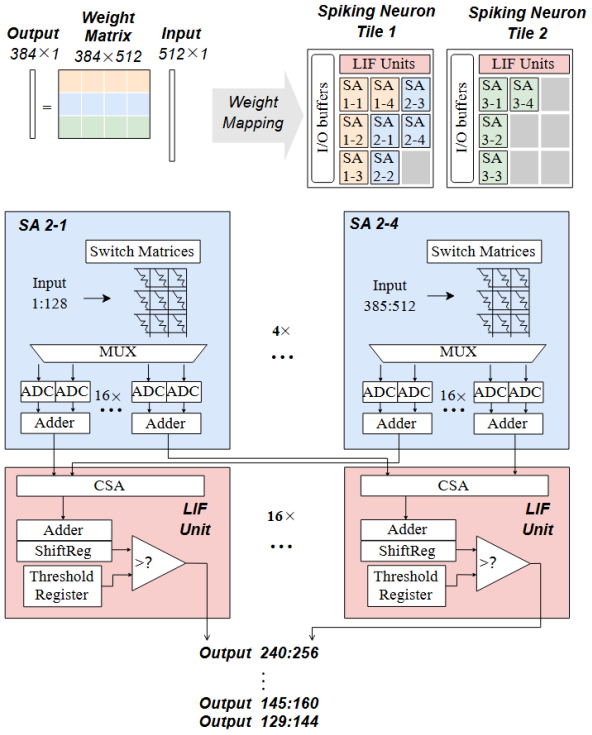

Figure 7 A illustration of the row-block-wise mapping strategy.

This mapping and accumulation approach is particularly advantageous for spike-based architectures, as it enables each tile to operate autonomously and locally without frequent accesses to global memory. It significantly reduces memory access latency, avoids bottlenecks, and leads to improved energy efficiency by keeping computation close to the data source.

3.4 SSA Engine: Hardware for Stochastic Spiking Attention

To overcome the inefficiencies of analog attention mechanisms in AIMC and general-purpose digital processors, the spiking transformer employs a hardware-efficient SSA engine. This engine implements multi-head self-attention entirely in the spiking domain, leveraging stochastic computing to replace conventional matrix multiplications with bitwise logic operations.

3.4.1 Stochastic Attention Tiles

The SSA engine is composed of multiple SSA tiles, each assigned to compute the attention for one attention head. As shown in Figure 8, each tile contains a $N \times N$ grid of stochastic attention cells (SACs), where N denotes the sequence length. This structure supports fully parallel computation of all pairwise query-key interactions required for attention score calculation. Each SAC computes one element of the attention score matrix S^t by receiving 1-bit spike-encoded data from the corresponding query and key vectors.

To compute the (i, j)-th element of the attention score matrix S^t, the SAC performs d_K bitwise AND operations between the iii-th row of Q^t and the j-th row of K^t, where d_K is the

feature dimension. The results are accumulated using a counter, and the accumulated sum is then used to generate a stochastic Bernoulli sample via a comparator. This sample becomes the attention score $S_{i,j}^t$, which is held in place for subsequent value weighting.

Each SAC is equipped with a local FIFO shift register to buffer the corresponding j-th row of V^t. During the value-weighting phase, the held $S_{i,j}^t$ is combined via bitwise AND with the buffered V^t, producing a local attention result. These outputs are summed across each column using a binary adder and passed through another Bernoulli sampling unit to yield elements of the final attention output matrix A^t. Because each SSA tile handles one attention head, the full multi-head attention output is obtained by instantiating multiple tiles operating in parallel.

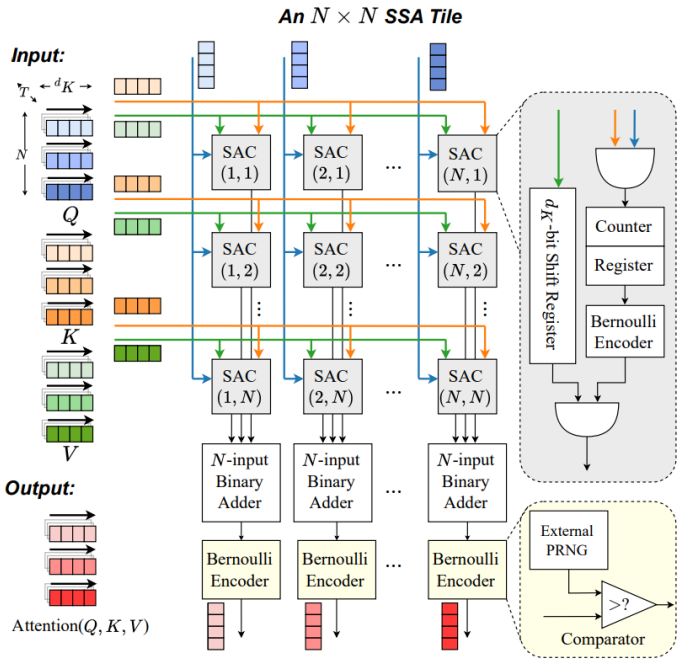

Figure 8 Block diagram of an N × N SSA tile.

3.4.2 Random Number Generation and Bernoulli Sampling

Bernoulli sampling within the SSA tiles is implemented using simple integer comparison. Rather than performing normalization of accumulated sums in analog or digital circuits, each accumulator output is compared to a uniformly distributed pseudo-random integer sampled from the interval $[0, I_{\max}]$, where $I_{\max} = d_K$ for attention scores and $I_{\max} = N$ for attention outputs. This avoids expensive division or floating-point operations. By choosing d_K and N as powers of two, hardware can exploit bit-width-aligned random values and fixed-point arithmetic.

Each SSA engine includes an LFSR-based PRNG array, supplying all required random numbers to the Bernoulli samplers across SACs. To maximize hardware utilization, a 32-bit LFSR is tapped byte-wise to generate multiple 8-bit random samples per clock cycle. This ensures that multiple SACs can be served simultaneously without requiring additional RNG hardware.

3.4.3 Streaming Dataflow and Pipelining

During inference, Q^t, K^t, and V^t are streamed into the SSA engine in a matrix-wise, time-serial manner. Each matrix is transmitted column by column over d_K cycles. The SACs perform one logical AND operation per cycle and accumulate results internally. After d_K cycles, attention scores S^t are generated in parallel. Value weighting proceeds immediately in the next d_K cycles, using the held scores and buffered values to compute the attention output A^t. The dataflow is shown in Figure 9.

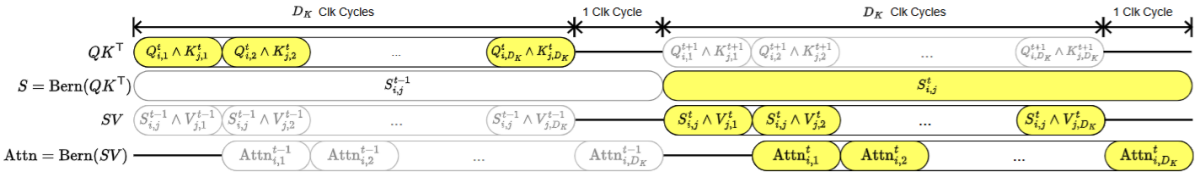

Figure 9 Illustration of dataflow design for the attention operation of each (i, j)-th SSA tile.

This streaming approach eliminates the need to store intermediate matrices globally, allowing the SSA engine to operate in a pipelined fashion. Each tile starts processing a new time step as soon as the previous one finishes. Due to the stateless design of SACs and LFSRs, SSA tiles can be reused across layers, minimizing area overhead.

The alternating execution of AIMC and SSA engines is coordinated via shared on-chip SRAM buffers. This buffer stores spike-encoded sequences between layers and ensures smooth data handoff between the AIMC and SSA processing phases, maintaining a continuous dataflow across the transformer blocks.

3.5 Hybrid Integration and System Design Considerations

The alternating roles of the AIMC and SSA engines are unified under a modular hybrid architecture designed to process spiking transformer workloads with high efficiency. This integration addresses both computational and memory transfer bottlenecks by distributing workload responsibilities between analog in-memory computing for static-weight matrix operations and logic-based stochastic computing for dynamic attention layers.

3.5.1 Temporal and Structural Modularity

The system operates under a time-multiplexed scheduling regime in which the AIMC engine processes static-weight feedforward and fully connected layers, while the SSA engine executes the multi-head attention computations. As shown in Figure 10, each engine is spatially separated and interconnected via on-chip shared SRAM buffers. This decoupling allows the engines to operate at their optimal timing granularity: token-wise streaming for the AIMC engine and matrix-wise streaming for the SSA engine.

This architectural modularity provides several advantages:

- **Pipeline continuity:** The event-driven execution is sustained without interruption by ping-pong buffering between SRAM banks, enabling overlap between data preparation and computation.

- **Layer reusability:** SSA tiles and LIF neuron logic are stateless across layers, which supports layer-wise reuse without additional reset or reconfiguration overhead.
- **Clock domain separation:** Each engine can be clocked independently based on its operational critical path and throughput requirement.

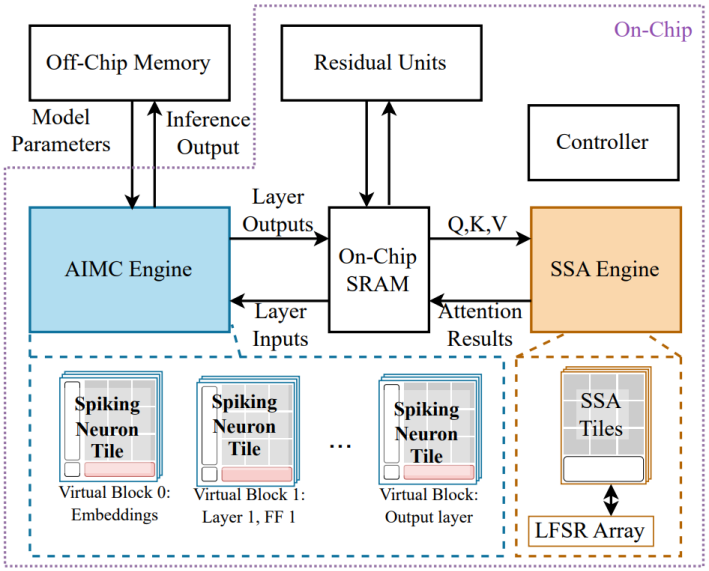

Figure 10 The overall hybrid analog-digital system architecture.

3.5.2 Data Flow Alignment and Spike Handling

The spike-based dataflow is carefully synchronized between the AIMC and SSA engines. As shown in Figure 11, in the AIMC engine, spike-encoded embeddings are propagated sequentially over T time steps, with membrane potentials in LIF units updated and reset at the granularity of individual tokens. Once the sequence of NNN token embeddings has been processed through the feedforward layers, the generated Q^t, K^t, and V^t embeddings are passed to the SSA engine.

The SSA engine expects full matrices at each time step and operates using column-wise streaming to compute the attention result. After generating the output matrix A^t, the attention-modulated embeddings are re-encoded as spike sequences and returned to SRAM. These are then processed by the next AIMC-based layer.

This separation of spike flow dynamics ensures compatibility between token-level processing and sequence-level attention without forcing uniform granularity across components. The design avoids storing membrane potentials across time steps for the SSA engine and eliminates the need for spike re-alignment or time expansion modules.

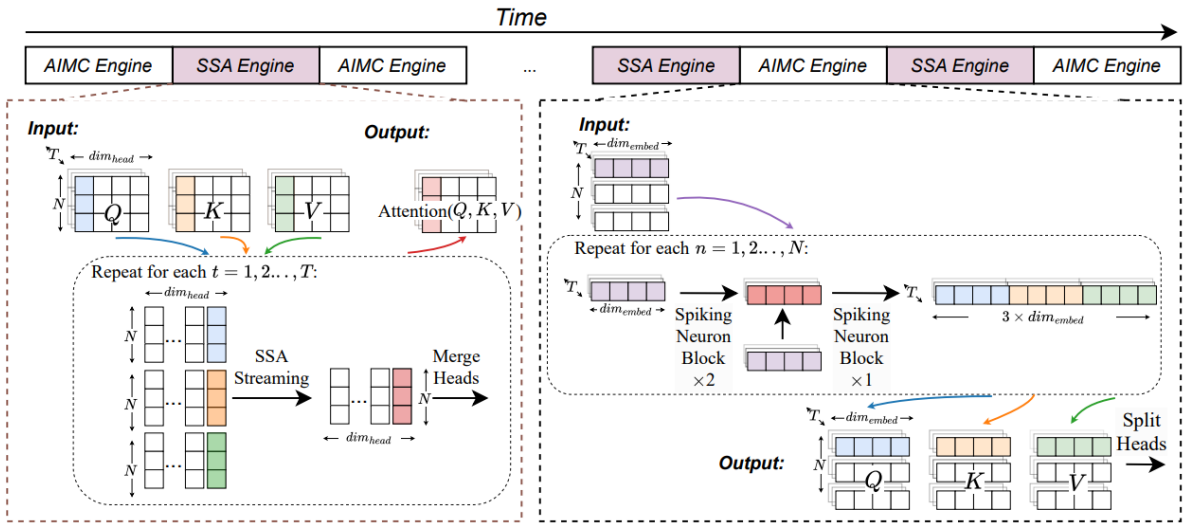

Figure 11 Illustration of the system inference dataflow.

3.5.3 Energy and Area Efficiency Profiling

The division of labor between analog and digital domains is optimized to minimize total energy and area consumption. Energy-intensive operations such as matrix-vector multiplications are offloaded to the AIMC engine, leveraging the constant-time execution and local accumulation of synaptic arrays. Meanwhile, attention logic, which is dynamic and non-static by nature, is executed by lightweight SSA logic using binary operations only, reducing the reliance on high-precision arithmetic or memory-intensive buffering.

From a hardware utilization standpoint, analog computation is distributed across many parallel columns in the AIMC crossbars, allowing efficient use of area and current flow. Digital logic for stochastic attention leverages bitwise sparsity and logical regularity, which allows synthesis with compact combinational logic blocks without requiring complex arithmetic units. Both the AIMC and SSA engines operate with shared SRAM and minimal communication interfaces between them, which helps to lower routing complexity and avoids the need for large shared caches or off-chip memory access. This hybrid analog-digital design achieves significant reductions in computational energy, control overhead, and processing latency. As a result, the architecture effectively meets the strict energy and real-time performance demands of edge AI signal processing in future wireless communication systems.

4 Spiking Neural Receiver: Training, Hardware Adaptation, and Demonstration

(This section introduces the training, calibration, and evaluation protocols developed in T2.2.4)

The spiking neural receiver introduced in Section 2 offers a biologically inspired solution to edge signal processing through ICL. By encoding pilot and data sequences as temporally sparse spikes, the model performs adaptive symbol detection without relying on explicit channel estimation. The use of a transformer backbone enables effective context reasoning over short sequences, while the spike-based formulation ensures compatibility with energy-efficient event-driven computation. Section 3 outlined a corresponding hardware design that implements this model on a hybrid architecture. AIMC is used to accelerate static-weight operations such as embedding and feedforward projection, while SSA provides a lightweight and hardware-native implementation of the attention mechanism. Together, these components form an integrated analog-digital substrate tailored to the demands of ICL-based neuromorphic inference.

This section focuses on completing the design-to-deployment pipeline. First, we describe how the ICL model is trained with hardware constraints in mind, including quantization, noise robustness, and spike rate control. Next, we detail how weights and neuron parameters are mapped to the AIMC and SSA engines and how the system is calibrated to account for device variation and signal timing. Finally, we present the full inference dataflow of the deployed receiver and evaluate its performance and energy efficiency under realistic MIMO channel conditions.

4.1 Hardware-Aware Training and Calibration for Analog Deployment

The training process for the spiking neural receiver follows a two-stage methodology. The first stage involves hardware-agnostic digital pre-training in an ideal simulation environment, where the network parameters are optimized without considering hardware-induced nonidealities. The second stage introduces hardware-aware fine-tuning using *AIHWKit*, incorporating statistical models of PCM variability, quantization, and drift to adapt the model to the real-world behavior of AIMC systems. Together, these stages prepare the network for robust deployment on the hybrid analog-digital hardware platform.

Following hardware-aware training, the spiking neural receiver requires system-level calibration techniques to maintain inference robustness under persistent nonidealities such as conductance drift, quantization noise, and device variability. This section details the runtime strategies implemented to mitigate performance degradation during deployment. These techniques, operating on top of the hardware-aware model trained in Section 4.1, ensure that energy-efficient and accurate inference remains feasible across extended operational lifespans.

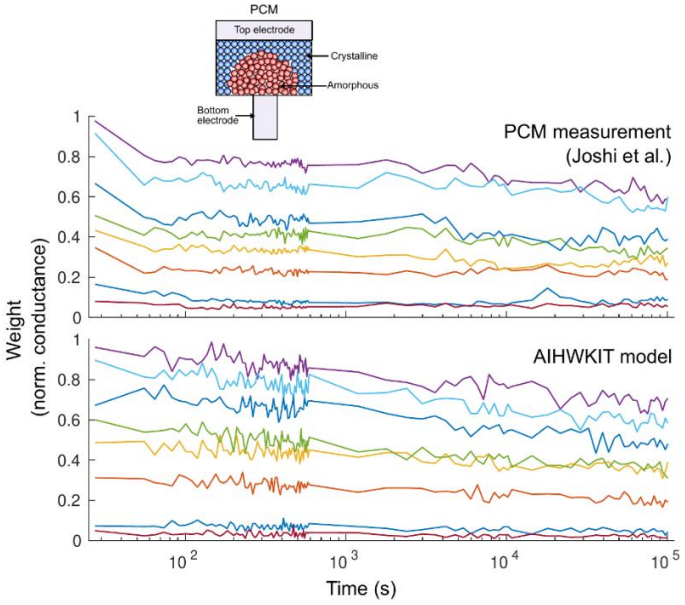

Figure 12 Comparison between experimentally measured PCM conductance drift (top, adapted from Joshi et al.) and the statistical drift model implemented in AIHWKit (Adapted from IBM AIHWKit documentation.). The plots show the normalized conductance values of multiple PCM cells over time, illustrating the long-term decay behavior and variability across devices. This agreement validates the realism of AIHWKit's drift modeling for use in hardware-aware training and simulation.

4.1.1 Modeling Drift and Noise in PCM Devices

Conductance drift is one of the most prominent sources of nonideality in PCM-based AIMC [18]. After initial programming, a PCM cell's conductance degrades over time following a power-law behavior. The conductance $G(t)$ evolves from its initial value G_0 according to the expression $G(t) = G_0 \left(\frac{t}{t_0}\right)^{-\nu}$, where \nu denotes the drift exponent and typically falls between 0.05 and 0.12. This drift reduces the synaptic current output for a given input voltage, resulting in progressively smaller matrix-vector multiplication outputs and ultimately impairing network inference.

As shown in Figure 12, the drift becomes particularly severe for high-exponent devices or longer post-programming intervals. In addition to drift, noise arises during write operations and due to cycle-to-cycle variability. Even under the same programming condition, the actual programmed conductance exhibits a distribution around the target value, modeled as Gaussian noise. Figure illustrates this deviation, emphasizing how random variation in write conductance accumulates over large matrix dimensions. Furthermore, the limited resolution of PCM introduces quantization effects. Each device supports only a fixed number of stable conductance states, typically around 16 levels in 4-bit encoding.

These nonidealities—drift, noise, and quantization—collectively degrade inference accuracy if left unaddressed. While *AIHWKit* simulates these effects during training, practical deployments demand dynamic calibration to correct the evolving hardware state.

4.1.2 Ideal Digital Pre-Training

The initial training is performed in a clean digital environment using the *SpikingJelly* framework. This stage establishes a strong baseline model for symbol detection using ICL. The spiking neural transformer is trained with LIF neurons, time window length T, 8-bit weight resolution, and QKV projections for multi-head self-attention. Input sequences are spike-encoded using Bernoulli sampling from their normalized amplitudes.

As detailed in Section 2.4, The training objective is to minimize cross entropy loss on the output token spikes, averaged across time steps. The context sequence $\mathcal{C}$ provides pilot labels for inference. The model outputs one-hot spike vectors per symbol class, which are compared with the ground truth to compute a standard cross-entropy loss.

BPTT is implemented using Sigmoid surrogate gradients. We use the AdamW optimizer with cosine learning rate decay and early stopping based on validation accuracy. All operations during this phase assume ideal, noise-free 8-bit computation.

This phase ensures that the network architecture and parameterization can solve the target MIMO detection tasks when unconstrained by hardware limitations. Once convergence is reached, the model parameters are exported for noise-injected hardware-aware adaptation.

4.1.3 Hardware-Aware Fine-Tuning with *AIHWKit*

To adapt the ideal model for physical deployment, we perform a second fine-tuning stage, referred to as hardware-aware training (HWAT), using *AIHWKit*, which simulates analog nonidealities of PCM-based AIMC. The pre-trained weights are quantized to 4-bit resolution and mapped to conductance values across the NVM crossbars. The following device-level imperfections are modeled during forward propagation:

- **Quantization and programming noise:** Limited resolution and stochastic write behavior are simulated during weight assignment.
- **Device-to-device variability:** Conductance samples for identical weight values vary spatially due to fabrication mismatches.
- **Conductance drift:** Temporal evolution of PCM conductance is captured using a logarithmic decay model.

During this phase, forward passes include stochastic perturbations to emulate the noisy behavior of analog memory during inference. However, gradients are computed under idealized assumptions to preserve training stability. This technique, known as noise-injected forward propagation, allows the model to learn a representation that is robust against noise while avoiding destabilizing the gradient flow.

We use the same loss function and optimizer as in the pre-training stage. The objective here is not to relearn the task but to adjust the weights within their feasible analog representation range to maintain accuracy in the presence of hardware-induced distortions.

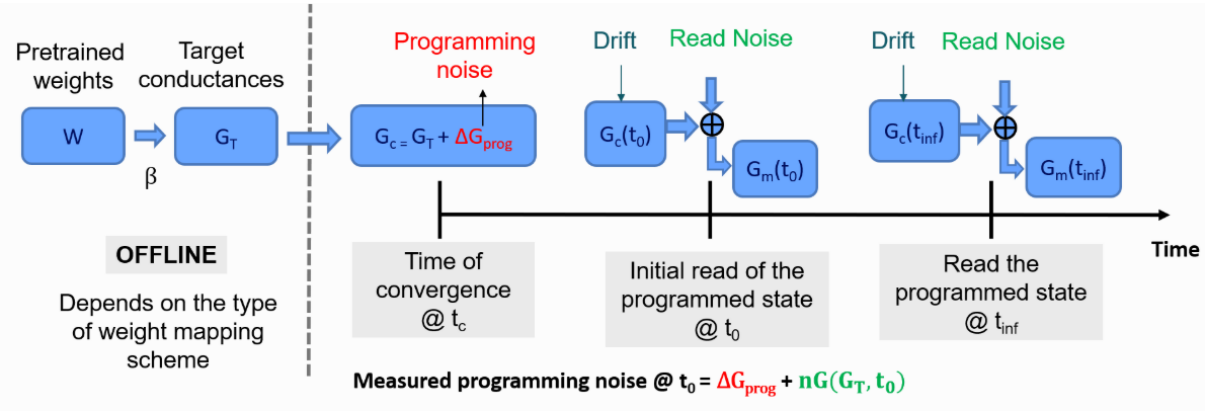

Figure 13 Statistical modeling of PCM weight degradation across time, as implemented in AIHWKit. The process starts with pretrained weights W mapped to target conductances G_T, followed by programming noise injection (ΔG_{prog}). Over time, conductance values drift and are further perturbed by read noise at each measurement point. The model reflects the cumulative effect of these nonidealities on inference, supporting accurate simulation of AIMC behavior during hardware-aware training. (Adapted from IBM AIHWKit documentation.)

Figure 13 referenced from the *AIHWKit* documentation support the realism of the simulated environment and validate the coverage of critical nonidealities during training.

4.1.4 Global Drift Compensation via Online Calibration

In spiking neural receivers, output information is encoded not just in value but also in time. The timing of spikes is crucial to inference performance, particularly in low-timestep configurations where spike sparsity is leveraged for energy efficiency. Conductance drift reduces output current, which in turn slows down the charging rate of the membrane potential in LIF neurons. As a result, the neuron may spike later than expected or fail to spike altogether, leading to temporal mismatches and information loss.

To address the runtime effects of drift without reprogramming weights, the spiking receiver employs global drift compensation (GDC). This technique estimates the scale at which synaptic outputs are reduced due to drift and compensates for the shift by applying a gain correction to all downstream outputs. The gain factor is estimated using calibration inputs periodically injected into the AIMC array. Known voltage patterns are applied, and the resulting output currents are compared with reference values stored during initial deployment. From this comparison, a scalar correction factor α is derived.

The scalar is applied to all subsequent outputs from the affected SA or tile, restoring their amplitude to the correct operational level. Since the calibration pattern is fixed and the operation is purely multiplicative, it can be performed by the system controller during idle periods with minimal latency. Figure 14 demonstrates the procedure of this calibration.

The gain factor may be applied at the granularity of individual SAs or groups of tiles, depending on the uniformity of drift observed across the hardware. The compensation mechanism is implemented with minimal overhead using on-chip accumulators and is compatible with shared ADC and LIF configurations used in the spiking neuron tiles.

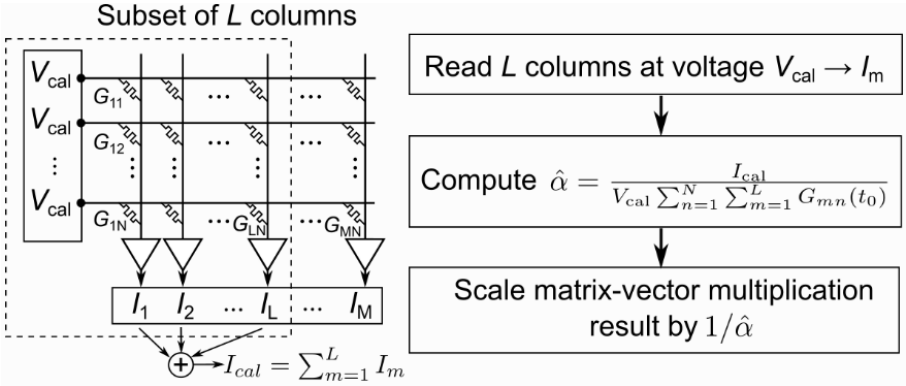

Figure 14 Global Drift Compensation (GDC) mechanism for PCM-based synaptic arrays. A known calibration voltage V_{cal} is applied to a subset of L columns, and the resulting current I_{cal} is measured. The gain correction factor $\hat{\alpha}$ is computed by comparing the measured current to the expected conductance sum, and this factor is used to rescale the matrix-vector multiplication output. This method corrects for drift-induced amplitude loss without requiring individual cell reprogramming. (Adapted from IBM AIHWKit documentation.)

4.2 Accuracy and Robustness Evaluation

We evaluate the inference accuracy and robustness of the spiking neural receiver under hardware constraints. The same ICL symbol detection task as conducted in Section 2.5. Results are obtained using a simulated hardware environment incorporating the analog nonidealities introduced in Section 4.1, using *AIHWKit* simulator.

4.2.1 Accuracy Evaluation

The ICL task follows the decoder-only formulation described in Section 2.3.2, where the receiver processes a sequence of pilot (query-answer) pairs and uses them to infer the correct label of a target symbol passed through an unknown MIMO channel. We evaluate the model under two antenna configurations—2×2 and 4×4 MIMO—corresponding to different levels of task complexity.

The number of query-answer pairs is fixed at 18. Accuracy is quantified using the BER, with lower BER indicating higher classification accuracy. We compare the spiking receiver deployed on the proposed hybrid analog-digital hardware (**SNN-Hybrid**) against two baselines: ANN-Digital, a full-precision transformer executed on digital devices; and SNN-Digital, a software-based SNN transformer executed digitally. All models are INT8-quantized during testing.

Results are summarized in Table 2. In the 2×2 setting, both SNN-Hybrid and SNN-Digital achieve competitive BERs using similar spike encoding lengths. For example, the SNN-Hybrid (4-256) achieves a BER of 0.067 using 6 time steps, while the SNN-Digital of the same size yields 0.061 with the same number of steps. As the number of antennas increases to 4×4, the problem becomes more complex due to the exponential growth in symbol classes. In this setting, the BER of smaller models degrades significantly.

Notably, scaling the model size from 4-256 to 8-512 significantly improves accuracy for both SNN-Digital and SNN-Hybrid. However, the benefit is more pronounced in SNN-Hybrid, which achieves a 0.114 BER reduction with 6 fewer time steps, compared to 0.110 reduction with

only 3 fewer steps in SNN-Digital. This demonstrates that SNN-Hybrid benefits more from increased model capacity, likely due to its ability to average out hardware noise and stochastic variations more effectively when equipped with higher-dimensional internal representations.

Table 2 Accuracy Performance on In-Context Learning for Wireless Symbol Detection

Model	Size Depth-Dim	2 × 2 Antennas BER (T)	4 × 4 Antennas BER (T)
ANN-Digital	4-256	0.051	0.141
	8-512	0.049	0.077
SNN-Digital	4-256	0.061 (6)	0.196 (7)
	8-512	0.058 (4)	0.086 (4)
SNN-Hybrid	4-256	0.067 (6)	0.205 (11)
	8-512	0.063 (4)	0.091 (5)

4.2.2 Robustness under Long-Term Conductance Drift

To assess the long-term robustness of the spiking receiver on the proposed hybrid analog-digital hardware platform (hereafter referred to as SNN-Hybrid), we evaluate its performance under conductance drift using the *AIHWKit* drift model described in Section 4.1.1. We simulate one year of drift in PCM devices and track BER degradation over time across different training and calibration strategies.

We consider four combinations of training and compensation: (i) conventional training (CT) without drift correction; (ii) CT with GDC; (iii) HWAT without GDC; and (iv) HWAT combined with GDC. In all cases, we observe the long-term trend of accuracy decay under simulated deployment.

Table 3 Long-term BER on ICL Symbol Classification Task (4 × 4 Antennas)

Size	CT+NC	HWAT+NC	CT+GDC	HWAT+GDC
4-256	0.475 (+0.27)	0.460 (+0.255)	0.268 (+0.063)	0.224 (+0.019)
8-512	0.503 (+0.412)	0.511 (+0.420)	0.152 (+0.061)	0.106 (+0.015)

The results in Table 3 show that both CT+NC and HWAT+NC degrade significantly after one year, with final BER values of 0.503 and 0.511, respectively, for the 8-512 model. Adding GDC significantly improves long-term performance: CT+GDC results in a long-term BER of 0.152 (with an absolute BER increase of 0.061 in one year, compare to newly programmed hardware), while HWAT+GDC pushes it further to as low as 0.106, with only an absolute BER increase of 0.015 in one year. The 4-256 models show similar long-term performance. Under HWAT+GDC, its final BER remains as low as 0.224, with only a 0.019 increase compared to the drift-free case.

The results confirm that both HWAT and GDC strategies are critical for stable long-term inference. GDC alone yields substantial improvements by correcting for global conductance degradation, while HWAT provides a learned tolerance to low-level hardware noise. The combination of the two consistently delivers the lowest error rates across model sizes.

These findings emphasize the need to co-optimize model capacity, training strategy, and drift compensation to achieve durable deployment of neuromorphic receivers. With HWAT and periodic GDC updates, the spiking receiver maintains reliable symbol detection performance over time, even in the presence of physical hardware degradation.

4.3 Efficiency Evaluation

This section presents the runtime efficiency of the spiking neural receiver during ICL-based symbol detection, evaluated under realistic hardware settings incorporating AIMC and SSA modules. All results reported here are obtained from hardware-simulated inference, accounting for analog nonidealities such as conductance drift, quantization, and peripheral circuit overhead. We report computational and memory access energy, latency, area breakdown, and comparative performance against state-of-the-art transformer accelerators. The ICL task tested here uses the 4×4 MIMO configuration, representing a high-complexity detection scenario for real-world neuromorphic communication systems.

4.3.1 Runtime Energy Consumption

Energy consumption is broken down into two components: computational energy (including MAC or AC operations) and runtime memory access energy. SNN-Hybrid is evaluated against three baselines: (i) ANN-Quant, a state-of-the-art digital transformer accelerator with INT8 MACs; (ii) ANN-Quant+AIMC, which replaces MACs with PCM-based AIMC arrays; and (iii) SNN-Digi-Opt, a digital implementation of spiking transformers using integer arithmetic and masked additions.

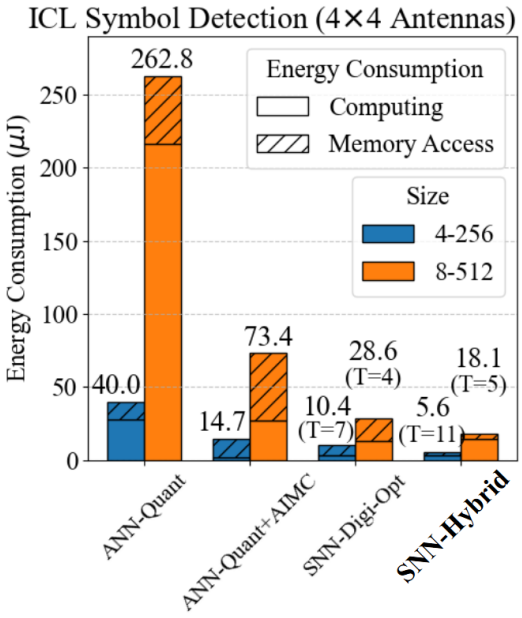

Figure 15 Energy consumption comparison between SNN-Hybrid and baseline implementations on the ICL symbol detection task (4 × 4 antennas).

All energy estimations assume INT8 precision for storage, uniform SRAM access, and preloaded weights in cache. Spike-based models are evaluated using the minimum required

encoding length T for convergence (Section 4.2.1). Hardware energy models are obtained using *DNN+NeuroSim V1.4* and synthesized with Cadence 45nm PDK for SSA.

Figure 15 compares the total energy consumption across baselines on the 4×4 antenna ICL task. SNN-Hybrid consistently outperforms all baselines. It achieves 7–14.5× lower energy than ANN-Quant, 2.6–4.0× lower than ANN-Quant+AIMC, and 1.6–1.8× lower than SNN-Digi-Opt. These savings are attributed to two main factors: the efficiency of in-memory MAC execution via AIMC, and the use of logic-gate-based SSA instead of integer multipliers.

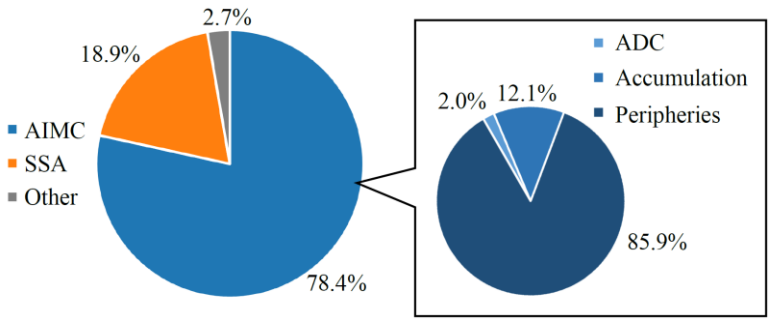

Figure 16 Breakdown of SNN-Hybrid computational energy.

Energy breakdown analysis (Figure 16) reveals that 78.4% of SNN-Hybrid computational energy is consumed by the AIMC engine, 18.9% by the SSA engine, and only 2.7% by residual units. Within the AIMC engine, 85.9% of energy is used by peripheral circuits such as decoders, MUXes, and buffers, while ADCs contribute just 2.0%. This is enabled by multiplexed ADC sharing (ratio of 8), which reduces static energy overheads. Accumulation circuits, including CSAs and LIF units, account for the remaining 12.1%.

Memory access energy is significantly lower than in ANN-based baselines. SNN-Hybrid avoids softmax operations and high-bit activations, and further reduces memory overhead by eliminating the need to store pre-activations through its row-block mapping strategy and SRAM-local dataflow. Additionally, the SSA engine performs all attention computations without intermediate storage, reducing total data movement.

4.3.2 Latency and Area Analysis

Figure 17 (left) presents the latency breakdown for one inference pass. More than 92% of the latency originates from peripheral operations, including data routing, buffering, and decoding. AIMC MVM latency is negligible (0.3%) due to constant-time crossbar execution. SSA logic, implemented using basic gates and counters, contributes just 2.0% to total latency.

In Figure 17 (right), SNN-Hybrid is compared to GPU-based inference using an ANN transformer and a spiking transformer implementation. Despite the extra temporal steps required by SNNs, SNN-Hybrid achieves a 2.18× speedup over the ANN baseline and a 6.85× speedup over the SNN-Digital implementation. These gains are due to efficient in-memory computing, stateless SSA execution, and avoidance of GPU-incompatible operations such as binary masking.

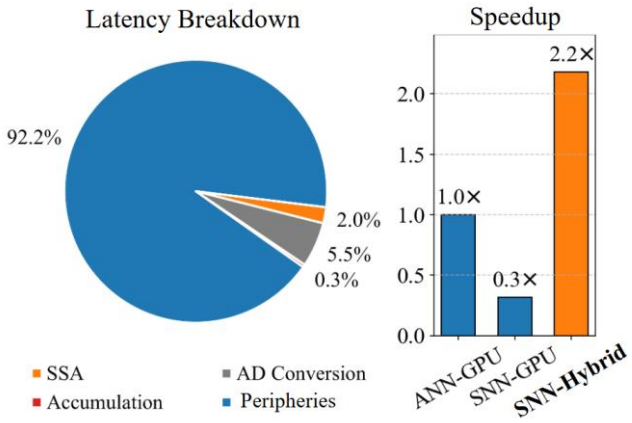

Figure 17 (Left) Latency breakdown of SNN-Hybrid and (Right) Latency comparison with ANN and SNN transformers on GPU.

Area estimation using *NeuroSim* and Cadence synthesis indicates a total area of 784 mm². The area breakdown is shown in Figure 18. The AIMC computing core (crossbars, ADCs, accumulators) occupies 11.5%, SSA tiles take 12%, and periphery and interconnect (IC) logic dominate with 76.5%. Compared to ALU-based architectures, the SSA engine achieves compact layout through simple control and logic gates.

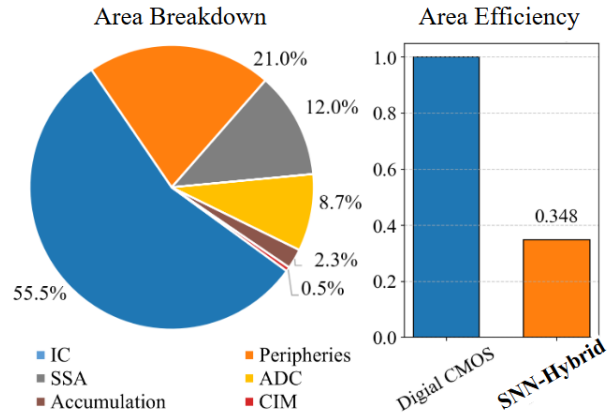

Figure 18 Chip area breakdown of the proposed hybrid analog-digital architecture.

4.3.3 Comparison with State-of-the-Art Accelerators`

Table 4 compares the proposed hybrid analog-digital architecture against SwiftTron [19] (digital ANN accelerator) and X-Former [20] (AIMC-based ANN transformer). On a normalized ICL benchmark, SNN-Hybrid consumes 13× less energy than SwiftTron and 6× less than X-Former. Latency is also improved due to SSA's lightweight design and the use of PCM multi-level cells, which reduce device count and memory footprint.

Notably, SwiftTron saves area via time-multiplexing, increasing runtime energy due to repeated memory accesses. X-Former stores attention intermediates in SRAM-based DIMC units, introducing latency overhead not present in SNN-Hybrid's logic-gate-based SSA flow.

Table 4 Comparison with SOTA Accelerators

Performance Metric	SwiftTron [19]	X-Former [20]	**Proposed** [21]
Computing Paradigm	ANN	ANN	SNN
MAC Implementation	Digital ALU	ReRAM-AIMC	PCM-AIMC
MHSA Implementation	Digital ALU	DIMC	SSA
Technology	65 nm	32 nm	45 nm
Weight Precision	INT8	INT8 (Equiv.)	INT5 (Equiv.)
Activation Precision	INT8/32	INT8	Multi-Step Binary
Frequency	143 MHz	200 MHz	200 MHz
Area (mm²)	273.0	–	784
Energy/Inference (mJ)	3.97	2.04	0.30
Latency/Inference (ms)	2.26	4.13†	2.18[i1]

4.3.4 Scalability

The proposed architecture is designed for scalable deployment. The SSA engine is stateless and modular, supporting reuse across attention heads and layers. Its 1-bit I/O buses minimize wiring complexity and ensure fast propagation. Meanwhile, the AIMC engine scales spatially via row-block-wise mapping and benefits from PCM's multi-level storage density. Compared to 1-bit ReRAM, PCM enables 3× fewer devices for the same precision, and similar cell sizes ($4F^2$–$8F^2$) support dense physical layouts [9].

For large-scale models, the architecture supports multi-tile scaling and chip-to-chip interconnects, enabling distributed deployment for MIMO scenarios beyond 4×4 antennas.

[1] The latency results for X-Former and X-Former account for both computational latency and data movement from memory to computing cores, while that for SwiftTron accounts only for computational latency.

5 Conclusion

This deliverable provides a complete design, implementation, and evaluation of a spiking neural receiver built to meet the energy efficiency and adaptability demands of future 6G wireless systems. By combining SNNs with a hybrid analog-digital hardware architecture, the system achieves accurate symbol detection under dynamic channel conditions while maintaining ultra-low energy consumption.

The report begins by establishing the theoretical foundation for neuromorphic computing and ICL, which enables selective and efficient processing of wireless signals. Based on this, a hardware architecture was developed that integrates analog AIMC cores for feedforward operations with a SSA engine tailored for binary spike-based inputs. The architecture avoids intermediate buffering and reduces data movement, resulting in a streamlined and scalable inference pipeline.

To ensure robustness on non-ideal analog hardware, a hardware-aware training framework was introduced. This approach includes quantization, noise, and conductance drift modeling during training. Calibration procedures, such as global drift compensation, further stabilize inference over time. Testing on simulated hardware confirmed that the receiver maintains strong BER performance on wireless ICL tasks, while consuming significantly less energy and incurring lower latency than conventional digital baselines.

Energy and latency analyses demonstrated that the receiver outperforms state-of-the-art ANN and SNN accelerators, especially under larger model configurations. The system also maintains long-term reliability, with minimal accuracy degradation over time, even under analog drift.

This work successfully completes the contributions set out in Tasks T2.1.2, T2.1.3, T2.2.3, and T2.2.4. It delivers a unified system that combines algorithmic design, hardware co-optimization, and calibration strategies into a viable neuromorphic transceiver platform suitable for edge deployment in real-world 6G communication environments.

References

[1] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural networks,* vol. 111, pp. 47--63, 2019.

[2] Z Song, O Simeone, B Rajendran, "Neuromorphic In-Context Learning for Energy-Efficient MIMO Symbol Detection," in *2024 IEEE 25th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Lucca, Italy, 2024.

[3] M. Zecchin, K. Yu, and O. Simeone, "In-Context Learning for MIMO Equalization Using Transformer-Based Sequence Models," in *2024 IEEE International Conference on Communications Workshops (ICC Workshops)*, Denver, CO, USA, 2024.

[4] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology,* vol. 15, p. 529–544, 2020.

[5] Z. Song, P. Katti, O. Simeone, and B. Rajendran, "Stochastic Spiking Attention: Accelerating Attention with Stochastic Computing in Spiking Networks," in *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)*, Abu Dhabi, United Arab Emirates, 2024.

[6] V Joshi, M Le Gallo, S Haefeli, I Boybat, SR Nandakumar, C Piveteau, M Dazzi, B Rajendran, A Sebastian, E Eleftheriou, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications,* vol. 11, p. 2473, 2020.

[7] W Fang, Y Chen, J Ding, Z Yu, T Masquelier, D Chen, L Huang, H Zhou, G Li, Y Tian, "Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence," *Science Advances,* vol. 9, p. eadi1480, 2023.

[8] MJ Rasch, D Moreda, T Gokmen, M Le Gallo, F Carta, C Goldberg, K El Maghraoui, A Sebastian, V Narayanan, "A Flexible and Fast PyTorch Toolkit for Simulating Training and Inference on Analog Crossbar Arrays," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Washington DC, DC, USA, 2021 .

[9] X Peng, S Huang, H Jiang, A Lu, S Yu, "DNN+ NeuroSim V2. 0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 40, pp. 2306 - 2319, 2020.

[10] A Shrestha, H Fang, Z Mei, DP Rider, Q Wu, Q Qiu, "A Survey on Neuromorphic Computing: Models and Hardware," *IEEE Circuits and Systems Magazine,* vol. 22, pp. 6 - 35, 2022.

[11] B. R. Gaines, "Stochastic computing systems," in *Advances in Information Systems Science*, Springer, 1969 , p. 37–172.

[12] A. Goldsmith, Wireless communications, Cambridge university press, 2005.

[13] Z Song, M Zecchin, B Rajendran, O Simeone, "Turbo-ICL: In-Context Learning-Based Turbo Equalization," *arXiv preprint arXiv:2505.06175.*

[14] Z Song, M Zecchin, B Rajendran, O Simeone, "In-Context Learned Equalization in Cell-Free Massive MIMO via State-Space Models," in *IEEE International Conference on Machine Learning for Communication and Networking*, Barcelona, Spain, 2025.

[15] S Garg, D Tsipras, PS Liang, G Valiant, "What can transformers learn in-context? a case study of simple function classes," *Advances in Neural Information Processing Systems,* vol. 35, pp. 30583--30598, 2022.

[16] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, I Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.

[17] Z Zhou, Y Zhu, C He, Y Wang, S Yan, Y Tian, L Yuan, "Spikformer: When spiking neural network meets transformer," in *The Eleventh International Conference on Learning Representations*, 2023.

[18] EJ Merced-Grafals, N Dávila, N Ge, RS Williams, JP Strachan, "Repeatable, accurate, and high speed multi-level programming of memristor 1T1R arrays for power efficient analog computing applications," *Nanotechnology,* vol. 27, 2016.

[19] A. Marchisio, D. Dura, M. Capra, M. Martina, G. Masera, and M. Shafique, "SwiftTron: An efficient hardware accelerator for quantized transformers," in *2023 International Joint Conference on Neural Networks (IJCNN)*, 2023.

[20] S. Sridharan, J. R. Stevens, K. Roy, and A. Raghunathan, "X-former: In-memory acceleration of transformers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 31, p. 1223–1233, 2023.

[21] Z Song, P Katti, O Simeone, B Rajendran, "Xpikeformer: Hybrid analog-digital hardware acceleration for spiking transformers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 33, pp. 1596 - 1609, 2025.

[i]