MindCore: Spike-Driven Programmable Accelerator for On-device Neuromorphic Computing

Hawon Park, Si Yong Lee, Ryangjin Lee, Yoora Kim
Department of Computer Science
Stony Brook University, NY, USA
{hawon.park, siyong.lee,
ryangjin.lee, yoora.kim}@stonybrook.edu

Yoon Seok Yang
Department of Computer Science
State University of New York Korea,
Incheon, Republic of Korea
yoonseok.yang@sunykorea.ac.kr

***Abstract*—Spiking neural networks (SNNs) are drawing attention as a promising way to achieve low-power, real-time intelligence on edge devices. In this work, we introduce MindCore, a spike-driven accelerator designed to support on-device neuromorphic computing. MindCore builds on a streamlined Reduced Instruction Set Compute (RISC) processor and provides programmable neurons and synapses, allowing flexibility across a wide range of SNN models. To handle spike-based operations efficiently, we developed a set of custom instructions, including Single Instruction Multiple Data (SIMD) accumulators for 8-, 16-, and 32-bit operations, and a spike-vector instruction that performs logical AND, spike counting, and accumulation in one step. The proposed design was implemented on a Xilinx ZCU104 FPGA and verified through Register Transfer Level (RTL) testing, together with a software toolchain for model compilation and application support. Demonstrations on vision inference and dementia diagnosis assistance confirm that MindCore achieves meaningful energy savings without sacrificing accuracy, showing its potential as a practical platform for future on-device neuromorphic AI.**

***Index Terms*—Spiking Neural Networks, Neuromorphic Computing, On-device AI, Custom Instruction Set Architecture, Spike Accumulators, Neuromorphic Circuits and Systems.**

I. Introduction

The widespread adoption of artificial intelligence has created a strong demand for computation at the edge, where devices must operate with strict constraints on power, size, and cost [1]–[3]. Conventional deep neural networks have shown remarkable success in applications such as computer vision and healthcare, but their heavy computational requirements make them difficult to deploy in portable or resource-limited systems. Neuromorphic computing, and in particular spiking neural networks (SNNs), has been explored as a more efficient alternative. By processing information in an event-driven manner, SNNs can reduce unnecessary computation and deliver low-latency responses, while naturally matching the temporal characteristics of biological signals [4], [5]. However, practical deployment of SNNs remains challenging, since most existing processors are not optimized for spike-based operations, and specialized hardware support is still limited [6].

To address this challenge, we propose MindCore, a spike-driven accelerator designed specifically for on-device SNN processing. The MindCore architecture is centered on an optimized Reduced Instruction Set Compute (RISC)-based controller [7], [8] that manages programmable neurons and synapses, enabling different models and classification tasks to be mapped efficiently. This provides flexibility and programmability for various SNN models and applications. The key novelty lies in its customized instruction set, which includes a general purpose arithmetic logic unit and Single Instruction Multiple Data (SIMD) accumulators [9] for multiple precision formats and a spike-vector instruction that executes bitwise AND, spike counting, and accumulation in a single operation. These instructions directly accelerate spike-driven convolution and spike-based self-attention, which are essential for emerging neuromorphic models. We also propose a hardware-based compression scheme for spike and event data that achieves a reduction of over 98% in volume, significantly minimizing memory and bandwidth requirements.

The architecture has been implemented and validated on a Xilinx ZCU104 FPGA, with a complete software stack for compilation, APIs, and application-level support. We demonstrate the use of MindCore in vision inference tasks and as a tool for dementia diagnosis assistance, showing that the accelerator can provide significant power reductions while maintaining accuracy. Our results suggest that MindCore is a step toward practical, scalable neuromorphic platforms capable of bringing energy-efficient intelligence to a variety of edge applications. The key contributions of this paper are as follows:

- **A Novel Accelerator Architecture:** We propose a novel spike-driven accelerator architecture, MindCore, featuring a specialized controller and programmable cores optimized for low-power and high computational efficiency, confirming its suitability for on-device AI.
- **A Custom Instruction Set for Spikes:** We develop a custom instruction set architecture (ISA) tailored for spike-based operations, including a dedicated instruction that uniquely accelerates spike-vector mechanisms.
- **Efficient On-Chip Data Compression:** The proposed spike compression method achieves over 98% data reduction, compressing 600 KB of event-stream input down to 6 KB, thereby greatly lowering on-chip memory and bandwidth requirements.
- **FPGA-Based Prototyping and Validation:** We present the full RTL implementation and FPGA-based validation

of the MindCore architecture on a Xilinx ZCU104 platform, demonstrating its practical feasibility.
- **Experimental Validation of Efficiency:** We demonstrate the system's real-world applicability through a complete software toolchain and representative applications in event-based computer vision and biomedical inference.

II. Related Work

Neuromorphic processors have been developed with a wide range of design philosophies, reflecting different priorities such as biological fidelity, programmability, and energy efficiency. One of the earliest large-scale digital efforts was IBM's TrueNorth chip [10], which integrated one million neurons and 256 million synapses while consuming only tens of milliwatts of power. Intel's Loihi [11] and Loihi 2 [12] advanced the field by emphasizing programmability and on-chip plasticity. These processors are among the first to show how architectural programmability and custom communication primitives can make SNNs more practical for a wide range of machine learning problems. Still, their instruction sets remain oriented around microcoded neuron updates and spike routing rather than providing single composite instructions for complex spike operations.

The BrainScaleS-2 [13] platform took a different path, employing mixed-signal analog circuits to accelerate neural dynamics and plasticity at speeds far beyond real time. However, its mixed-signal nature and scale-oriented design make it less suited for portable, power-constrained deployments. Similarly, SpiNNaker2 [14] represents another architectural extreme, leveraging millions of ARM cores in an event-driven communication network. This architecture prioritizes flexibility and scalability for large-scale neural simulations, but as a software-programmable platform, its per-event energy is higher than that of custom digital or analog neuromorphic accelerators.

More recently, the Chinese research community has presented Darwin3 [15], a digital neuromorphic chip designed around a domain-specific instruction set. Darwin3 introduces more than ten spike-centric custom instructions, compressed connectivity schemes, and on-chip learning support. Darwin3 illustrates the growing interest in ISA-level innovation for neuromorphic computing, though its published results emphasize architectural scale and mapping benchmarks rather than low-power edge inference.

At the edge, commercial and academic groups have begun introducing neuromorphic processors that explicitly target on-device AI. BrainChip's Akida processor [16] integrates event-based kernels and hybrid spike/CNN support, exposing instruction-like spike primitives that allow convolutional and spiking workloads to be executed on the same platform under tight power budgets. In parallel, Zhou et al. [17] reported a heterogeneous neuromorphic SoC in 55 nm CMOS that couples a RISC-V host with neuromorphic cores and extends the ISA with spike update instructions.

As summarized in Table I, recent neuromorphic processors vary widely in their computational models and instruction-level design. While TrueNorth and BrainScaleS-2 emphasize biological fidelity and power efficiency, they lack ISA-level programmability. In contrast, MindCore introduces a dedicated instruction for composite spike-vector operations, offering fine-grained spike processing acceleration that has not been explicitly implemented in prior ISA-based designs and optimizing spike-based acceleration using the customized instructions while meeting the programmable requirement. This is the unique contribution of MindCore. Alongside its SIMD 8/16/32-bit accumulators, MindCore introduces a composite spike-vector instruction that shortens the critical path for spike-driven convolution and self-attention. By embedding this capability directly in the ISA, MindCore provides a fast and energy-efficient data path for on-device neuromorphic inference, addressing an open gap left by prior designs.

III. MindCore Chip Design: Spike-Driven Programmable Accelerator for Embedded Neuromorphic Systems

The MindCore architecture was designed with the goal of enabling efficient spike-based computation in embedded neuromorphic platforms. The baseline design adopts the controller and data path structure from a RISC-based processor, which were extensively modified and optimized for event-driven processing. The resulting architecture integrates a dedicated *Spike Processing Unit (SPU)* that handles spike-based arithmetic and communication with minimal control overhead.

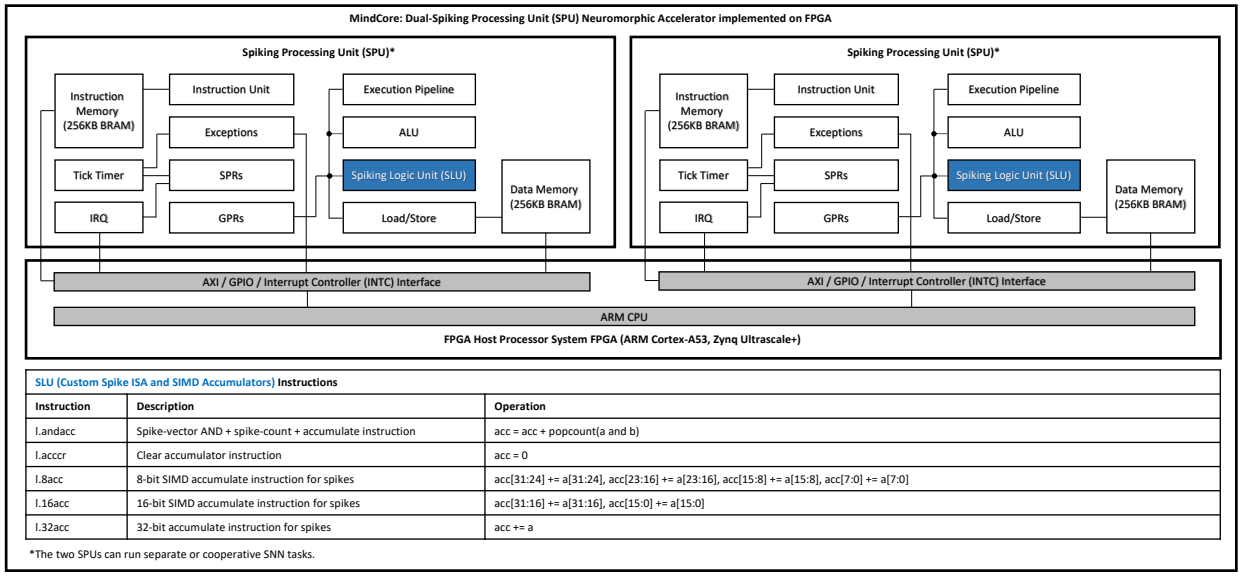
A. RISC-Based Controller and Spike Processing Unit

The MindCore controller follows a streamlined RISC pipeline, responsible for instruction scheduling, memory access, and spike data management. To accommodate the temporal and sparse characteristics of SNNs, the instruction pipeline and register file were reorganized to minimize unnecessary switching activity and idle cycles. The data path was extended to include the SPU, which executes spike-domain operations directly in hardware. The SPU supports programmable neuron and synapse models while maintaining compatibility with standard arithmetic and control instructions, ensuring flexibility for diverse applications.

The overall architecture of the proposed MindCore system is shown in Fig. 1. The design consists of two Spiking Processing Units (SPUs), each integrating an instruction unit, Arithmetic Logic Unit (ALU), load/store unit, and a dedicated Spiking Logic Unit (SLU) for spike-based operations. The SPUs share a common ARM host through an Advanced eXtensible Interface (AXI) and can operate independently or cooperatively depending on the assigned task. Each SPU contains 256 KB of instruction and data memory, Special-Purpose Registers (SPRs), and General-Purpose Registers (GPRs) for instruction-level programmability. The SLU implements spike-oriented custom instructions that perform logical operations, spike counting, and SIMD accumulation within a single cycle. This structure allows MindCore to process spiking workloads efficiently while maintaining compatibility with the RISC

TABLE I: Comparison of Representative Neuromorphic Processors (ISA-level Spike Support Highlighted).

Platform	Programming / Learning	ISA / Custom Operations	Scale / Integration	Focus and Notes
IBM TrueNorth [10]	Offline learning	Fixed microcode (no ISA-level SNN ops)	1M neurons / 256M synapses	Ultra-low static power; limited adaptability.
Intel Loihi [11] / Loihi 2 [12]	Programmable neurons; on-chip learning	Microcoded neuron updates and routing primitives	Many-core mesh	Programmable plasticity; not explicit ISA-level SNN ops.
BrainScaleS-2 [13]	Analog accelerated dynamics	Analog plasticity configuration (not ISA-level)	Wafer-scale mixed-signal system	High biological fidelity; not ISA-driven.
SpiNNaker2 [14]	Software-programmable	Standard ARM ISA; software-based SNN libraries	Millions of ARM cores	Highly flexible, but lacks custom spike ISA extensions.
Darwin3 [15]	Programmable neurons; on-chip learning	**Domain-specific ISA with SNN instructions**	Multi-million neurons	Improved mapping efficiency; large-scale spiking workloads.
Akida (BrainChip) [16]	On-chip learning; hybrid SNN+CNN	**Event-based instruction primitives**	Commercial SoC; edge AI focus	Instruction-like spike kernels for embedded applications.
Zhou et al. SoC [17]	Hybrid SNN updates; edge tasks	**RISC-V ISA extensions for SNN operations**	55 nm CMOS; 0.96 pJ/SOP	Edge neuromorphic SoC with ISA-level spike processing.
MindCore (this work)	Programmable neurons; edge inference	**SIMD accumulators; spike-vector AND+count+accumulate**	FPGA (ZCU104) prototype	Unique ISA-level composite spike-vector operation for fast, low-power on-device SNN inference.



SLU (Custom Spike ISA and SIMD Accumulators) Instructions

Instruction	Description	Operation
l.andacc	Spike-vector AND + spike-count + accumulate instruction	acc = acc + popcount(a and b)
l.acccr	Clear accumulator instruction	acc = 0
l.8acc	8-bit SIMD accumulate instruction for spikes	acc[31:24] += a[31:24], acc[23:16] += a[23:16], acc[15:8] += a[15:8], acc[7:0] += a[7:0]
l.16acc	16-bit SIMD accumulate instruction for spikes	acc[31:16] += a[31:16], acc[15:0] += a[15:0]
l.32acc	32-bit accumulate instruction for spikes	acc += a

*The two SPUs can run separate or cooperative SNN tasks.

Fig. 1: Block diagram of the MindCore dual-spiking processing unit (SPU) neuromorphic accelerator implemented on a Xilinx ZCU104 FPGA. Each SPU includes programmable instruction and data memories, a RISC-based controller, and a custom Spiking Logic Unit (SLU) that supports SIMD spike accumulation and composite spike-vector operations through dedicated ISA extensions. All the SIMD and spike-vector instructions execute in a single cycle within the SLU, allowing concurrent spike accumulation across multiple bit-lanes.

execution pipelines. The FPGA supports real-time event-driven inference under low-power and moderate resource utilization.

B. Custom ISA for Spike Computation

A key aspect of the MindCore design is the inclusion of custom spike-oriented instructions that accelerate key SNN operations at the hardware level as shown in Fig. 1. These include SIMD accumulators supporting 8-, 16-, and 32-bit precision formats, and a composite *spike-vector instruction* that performs bitwise AND, spike counting (i.e., pop counting), and accumulation in a single cycle. This instruction set enables fast spike-based convolution and self-attention, both of which are computational bottlenecks in modern neuromorphic workloads. The instructions were implemented through dedicated datapaths and ALU extensions, ensuring low latency and full synthesizability in standard RTL design flows.

C. Efficient Event Compression and Memory Optimization

Compression/Decompression modules are incorporated into the SPU in order to address bandwidth and memory challenges for datasets whose sample sizes exceed the memory bandwidth allowed by MindCore. For example, one N-MNIST sample requires around 600KB of memory in INT8 space. The compression module reduces memory usage to an average of 6 KB with a maximum of 10 KB. This corresponds to a reduction of more than 98% in data volume. Table II shows the full performance metrics of applying our compression module.

TABLE II: Performance Metrics of RLE+VLQ Compression

Metric	Original Dataset	MaxPool Dataset
Initial Size (Bytes)	614,400	173,400
Avg. Compressed Size (Bytes)	6,684	6,684
Max Compressed Size (Bytes)	10,566	10,566
Avg. Size Reduction	**98.91%**	**96.15%**
Reduction at Max Size	98.28	93.91%
Avg. Compression Rate	91.93x	25.94x
Rate at Max Size	58.15x	16.41x

The compression module itself is a combination of Run-Length Encoding (RLE) [18] and Variable-Length Quantity

(VLQ) Encoding [19]. RLE is a lossless compression algorithm that replaces consecutive sequences, or runs, of identical symbols with the symbol and a count of its length. Note that RLE is only useful if the memory required to store the compressed data is less than the original data size. This means that RLE is optimized for highly sparse inputs such as N-MNIST, which has an average sparsity of 97.97%. VLQ is used in tandem to store the large number of runs generated by RLE. For example, instead of allocating 32 bits per run length, we try to encode the run lengths so that smaller numbers take up less space. This is necessary due the wide distribution in run lengths per N-MNIST timestep. While the average run length of a N-MNIST sample is 28.67, the minimum run length is 1 and the maximum run length is 578. Naively storing these run lengths would result in wasted space. VLQ is able to represent a non-negative number V as a sequence of bytes represented in base 128.

Alg. 1 and Alg. 2 describe the overall encoding and decoding process.

Algorithm 1 Overall Compression Process

procedure CompressStream(S)
Require: Raw bit sequence S
Ensure: Compressed byte stream B
  $(start\_bit, L) \leftarrow$ RunLengthEncode*(S)
  $bit\_list \leftarrow [start\_bit]$
  for each length l in L **do**
    $vql\_bytes \leftarrow$ VLQ_Encode**(l)
    Append bits of vql_bytes to bit_list
  end for
  $B \leftarrow$ PackBitsToBytes(bit_list)
  return B
end procedure

*RunLengthEncode procedure was inspired by Birajdar et al. [18].
**VLQ_Encode procedure was inspired by Li et al. [19].

Algorithm 2 Overall Decompression Process

procedure DecompressStream(B)
Require: Compressed byte stream B
Ensure: Decoded bit sequence S_{dec}
  $bit\_iter \leftarrow$ GetBitIterator(B)
  $start\_bit \leftarrow$ NextBit(bit_iter)
  $L \leftarrow []$
  while bit_iter is not empty **do**
    $l \leftarrow$ VLQ_Decode**(bit_iter)
    Append l to L
  end while
  $S_{dec} \leftarrow$ RunLengthDecode*$(start\_bit, L)$
  return S_{dec}
end procedure

*RunLengthDecode procedure was inspired by Birajdar et al. [18].
**VLQ_Decode procedure was inspired by Li et al. [19].

Altogether, this compression strategy allows MindCore to maintain efficient data throughput even under high event rates, making it suitable for real-time embedded inference. Neuromorphic datasets like N-MNIST are compressed across each timestep and individually decoded in the SPU. The datasets are examined further in section IV.

D. Programming Neurons and Synapses for Diverse SNN Models

In the MindCore framework, neurons and synapses are defined and configured through a programmable software model that interfaces directly with the SPU. This approach provides flexibility in defining both network topology and neuron dynamics while maintaining tight integration with the underlying instruction set and runtime control.

For the models presented in this work, the SNN configuration consisted of approximately 678 Leaky Integrate-and-Fire (LIF) neurons and 70621 synaptic connections. These parameters were selected to represent realistic, lightweight workloads for embedded neuromorphic inference. Each neuron is instantiated as a software-programmable entity, with parameters such as membrane potential, threshold, decay constants, and refractory behavior controlled through register-level configuration. Synaptic connections are stored in compressed tables that specify presynaptic and postsynaptic indices, weights, and delay terms. Table III shows the actual resource utilization of our selected models. Note that T-MNIST is a special MNIST model that uses ternary weight quantization and bit packing to efficiently store weights and activations.

TABLE III: Resource Utilization of Selected Models

Model	Precision	Neurons	Synapses	Memory (KB)
MNIST	FP32	986	109,386	427.29
	INT8	986	109,386	106.82
N-MNIST	FP32	780	83,018	324.29
	INT8	780	83.018	81.27
ADFTD (EEG)	FP32	268	19,459	76.01
	INT8	268	19,459	19.26
T-MNIST	INT8	986	109,386	27.36

The programmable interface also allows substitution of different neuron dynamics depending on the target model. In addition to standard LIF neurons, the MindCore framework supports adaptive LIF [20], Sigma–Delta [21], [22], and Hodgkin–Huxley [23], [24] neuron types, each of which can be configured with distinct time constants and feedback parameters. This flexibility enables direct experimentation with biophysically inspired models as well as computationally efficient approximations.

Similarly, the modular synapse representation allows implementation of various network topologies, ranging from convolutional spiking neural networks (CSNNs) to transformer-based architectures. The system supports hierarchical layering and recurrent spike feedback, allowing the same hardware to execute convolutional, feed-forward, or attention-based spiking models with minor software adaptation. These capabilities make MindCore applicable across a wide spectrum of SNN research, from event-based vision tasks to neuromorphic signal processing and biomedical applications.

E. Model Design

1) LIF Neurons: The LIF Neuron is one of the most commonly used spiking neuron models [25]. The LIF neuron models a biological neuron's cell membrane as a Resistor–Capacitor (RC) circuit.

2) SNN Model Architecture: MindCore's fully programmable nature allows for any model to be loaded as long as it can fit inside 256KB. Table IV presents three fully-connected SNN architectures that use the LIF neurons.

TABLE IV: Model Architecture for Selected Models in MindCore

Dataset	Input	Hidden1	Hidden2	Output
MNIST	784	128	64	10
N-MNIST	584	128	64	10
ADFTD (EEG)	9	128	128	3

3) Integer Quantization: Integer quantization is the process of mapping floating-point numbers to a constrained set of integer values [26]. While integer quantization is primarily used to reduce the computational and memory costs of deep learning networks, it is a necessity for MindCore which does not support floating point logic. This means that our models require some postprocessing to translate the full precision weights and activations into integers.

Eq. 1 defines the basic quantization function where r is an arbitrary real number, S is a scale factor, and Z is a zero point offset. q_{min} and q_{max} refer to the integer range that r is constrained to.

$$q = \text{clip}\left(\text{round}\left(\frac{r}{S}\right) + Z, q_{\text{min}}, q_{\text{max}},\right) \tag{1}$$

The standard process is to apply symmetric quantization on the weights and asymmetric quantization on the inputs and activations [27], [28]. MindCore, however, implements a simpler INT8 quantization scheme that uses a global scale factor in lieu of dynamically calculated scale factors as in symmetric/asymmetric quantization. Table V shows the quantization parameters used quantize the model's weights and biases into INT8 representation.

TABLE V: INT8 Quantization Parameters for Selected MindCore Models

Model	Shift	Scale	q_{min}	q_{max}
MNIST	3	8	-128	127
N-MNIST	4	16	-128	127
ADFTD (EEG)	4	16	-128	127
T-MNIST	4	16	-128	127

4) Ternary Quantization: Ternary quantization is an extreme form of integer quantization that limits the integer range to a set of $\{+1, 0, -1\}$ [29]. This is achieved by using a threshold Δ. A scale factor α is also computed which is used to approximate the original magnitude of the quantized tensor.

$$w_t = \begin{cases} +1 & \text{if } w > \Delta \\ -1 & \text{if } w < -\Delta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Eqn. 2 gives the quantization function as described above. We can then approximate some tensor value $W \approx \alpha W_t$ where α is a computed, real-valued scalar value.

5) Removing MAC using popcount32: Multiply-And-Accumulate (MAC) operations are known to be the bottleneck for neural networks. By using integer quantization, we can replace these expensive MAC operations with sparse additions. Instead of multiplying an input vector by some weight matrix, we can simply just accumulate the weights for nonzero weights as shown by Eqn. 3. This significantly reduces computational overhead and memory usage for highly sparse binary inputs.

$$c_{ij} = a_{ij} + b_{ij} \quad \text{for } i = 1, \ldots, m \text{ and } j = 1, \ldots, n \tag{3}$$

Note that we can take this even further with ternary weights by using population count (popcount) [30]. The popcount of n is the sum of its binary bits as defined by Eqn. 4.

$$\text{popcount}(n) = \sum_{i=0}^{k-1} b_i \tag{4}$$

However, popcount requires bitpacked binary inputs. This means that we need to split our ternary weight W_t into two separate binary tensors W_{pos} and W_{neg}. These two tensors are masks that indicate the positions of the $+1$ and -1 values in the original ternary weight matrix.

$$W_{pos} = I(W_t = 1) \quad \text{and} \quad W_{neg} = I(W_t = -1) \tag{5}$$

$$Y = \sum_i A_i \cdot W_{pos,i} - \sum_i A_i \cdot W_{neg,i} \tag{6}$$

Eqn. 5 shows how to define the two binary tensors W_{pos} and W_{neg}. Eqn. 6 defines the MAC operation between a binary input A and the binary weight matrices W_{pos} and W_{neg}. Since both the inputs and weights are binary, the MAC is simply the number of indices where both vectors are equal to 1. We can efficiently do this with a bitwise AND of the bitpacked vectors and then a popcount operation.

$$Y = \text{popcount}(A' \ \& \ W'_{pos}) - \text{popcount}(A' \ \& \ W'_{neg}) \tag{7}$$

Eqn. 7 can be used to replace a series of expensive MACs with just two bitwise ANDs and two popcount instructions which drastically reduces computational load and memory usage. We can then scale the popcount output by the precomputed α value to approximate the original floating-point output.

IV. Experiments and Results

All functional blocks of the MindCore system were described in fully synthesizable RTL and mapped onto a Xilinx ZCU104 FPGA platform for real-time validation. The synthesized design was evaluated for timing closure, resource utilization, and dynamic power consumption. Measured power data were compared with those of a desktop GPU and an embedded GPU under equivalent workloads, confirming that the proposed spike accelerator achieves substantial energy savings. The average operating power was significantly lower, demonstrating the efficiency of event-driven computation in hardware.

For benchmarking and verification, we implemented a complete toolchain for compiling SNN models and loading spike event data into the FPGA runtime environment. The evaluation covered both vision and biomedical applications. Benchmark datasets included MNIST [31], N-MNIST [32], and ADFTD [33], a biomedical dataset for Alzheimer's and dementia diagnosis with electroencephalography (EEG). Across all tests, MindCore consistently achieved efficient on-device inference with minimal power consumption, validating its effectiveness as a programmable neuromorphic accelerator suitable for real-world embedded AI applications.

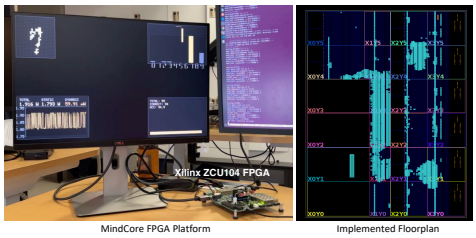


Fig. 2: MindCore FPGA prototype and implementation results. The left image shows the MindCore platform running on a Xilinx ZCU104 FPGA board with real-time spike activity and inference visualization. The right image presents the implemented floorplan, showing the placement and routing of dual-SPU and SLU modules on the FPGA fabric.

A. FPGA Prototyping

1) System Integration and Hardware Prototyping: The complete MindCore system was implemented and validated on a Xilinx ZCU104 evaluation platform featuring a Zynq UltraScale+ MPSoC. The design integrates the MindCore spike accelerator as a programmable logic (PL) subsystem, tightly coupled with the PS through AXI interconnects. The SPU controller, event compression module, and on-chip memory were synthesized as independent RTL modules and interconnected using AXI4 interfaces for instruction and data exchange.

The MindCore dual-SPU FPGA prototype was synthesized using Xilinx Vivado with a target clock frequency of 100 MHz. Timing closure was achieved without the need for manual placement constraints, and resource utilization remained within 9% and 4% of the available Look Up Tables (LUTs) and Registers, respectively, and 82% of the BRAM resources as summarized in Table VI. These results confirm that the proposed architecture can realize efficient spike-based computation and custom ISA execution within a compact hardware footprint, demonstrating its practicality for low-power, on-device neuromorphic computing.

Fig. 2 presents executing real-time spike-based inference on the FPGA board with the visualization of classification results and the implemented floorplan after place-and-route. The floorplan shows the distribution of the dual Spiking Processing Units (SPUs) and BRAMs across the FPGA fabric.

Power analysis was conducted using Xilinx XPower Analyzer and direct current measurements through the board power monitor. The average active power remained below 1.9 W, while idle power measured at approximately 1.8 W, demonstrating the energy efficiency of the event-driven execution model.

TABLE VI: Resource Utilization of *MindCore* on Xilinx ZCU104 FPGA

Resources	LUTs (230,400)	Registers (460,800)	Block RAM Tile (312)	DSPs (1,728)
Utilization (Used%)	21,269 (9%)	18,161 (4%)	256.5 (82%)	8 (0.5%)

2) Software Environment and Toolchain Support: A complete software stack was developed to enable compilation, instruction generation, and runtime control for the MindCore architecture. The software flow includes four major stages: model conversion, instruction scheduling, binary generation, and host communication.

For model conversion, SNN models defined in PyTorch and CuPy were translated into MindCore-compatible graph descriptions through a custom Python front-end. The instruction scheduler then maps each layer to a set of MindCore custom instructions, assigning SIMD and spike-vector operations according to layer characteristics. The resulting binary files are transferred via the Processing System (PS) interface to the accelerator at runtime.

The runtime environment, implemented in C/C++, provides an API for loading models, managing spike data streams, and monitoring accelerator status. Through this interface, users can configure neuron parameters, threshold dynamics, and timing windows directly from the host processor. The driver layer also supports DMA-based data transfer between the PS memory and on-chip BRAM, allowing efficient spike data loading with minimal latency.

3) Dataset Preprocessing: Event datasets, by nature, contains many timesteps and can easily exceed the 256KB memory limit set by MindCore. N-MNIST is no exception and is too big to train and load into memory at its original resolution and timestep width. Thus, N-MNIST is first preprocessed with MaxPooling to reduce the input dimensions from $2\times32\times32$ to $2\times17\times17$. This new resolution allows for the trained weights and activations to fit within the MindCore memory limit. N-MNIST is then run through the compression algorithm. MNIST and ADFTD, on the other hand, do not require any extensive preprocessing. Poisson encoding is applied on MNIST to generate binary spikes and repeated for 100 timesteps. ADFTD (EEG) is just 19 channels of raw signals. As such, the only preprocessing required is to normalize each channel with its mean and standard deviation. Both MNIST and ADFTD do not require any additional compression as they fit well within the bounds of the MindCore memory limit.

TABLE VII: Training Hyperparameters for Selected MindCore Models

Model	τ	v_{th}	v_{reset}	τ_q	v_{th_q}	$\frac{1}{\tau_q}$
MNIST	2	1	0	16	8	4
N-MNIST	2	1	0	32	16	8
ADFTD (EEG)	2	1	0	32	16	8
T-MNIST	2	1	0	32	16	8

4) Model Training and Inference: Fig. 3 shows the overall training and inference pipeline. The models are trained on a NVIDIA RTX A5000 GPU for 100 epochs each. The highest performing weights are then extracted and quantized to INT8 and ternary respectively. Table VII shows the hyperparameters used during FP32 training and INT8 inference. To support ternary inference, three additional scaling values were used: $\alpha_1 = 0.24$, $\alpha_2 = 0.44$, and $\alpha_3 = 0.78$, which were quantized to 3, 7, and 12, respectively.

Inference on the Xilinx ZCU104 FPGA board follows a multi-stage hardware–software co-execution process. The ARM Cortex-A53 host first establishes communication with the SPU by initializing the general-purpose input/output (GPIO2) interface and AXI interrupt controller (INTC) triggers. The A53 then transfers the compiled SPU binaries to shared memory, performing byte-order conversion throughout the process. This conversion is essential because the A53 core

TABLE VIII: Benchmarking result

Dataset	Hardware	Precision	Latency (ms)	Total Power (W)	Static Power (W)	Dynamic Power (mW)
MNIST	RTX A5000*	FP32	**82.989**	66.414	13.284	53129.873
		INT8	153.538	69.041	13.284	55756.845
	Jetson Xavier AGX*	FP32	203.901	4.319	2.789	1529.639
		INT8	388.507	4.425	2.789	1636.412
	MindCore*	INT8	1110.799	**1.832**	**1.805**	**26.948**
N-MNIST	RTX A5000	FP32	**67.586**	66.354	13.284	53069.939
		INT8	306.361	70.95	13.284	57666.263
	Jetson Xavier AGX	FP32	171.104	4.321	2.789	1531.55
		INT8	784.417	4.339	2.789	1550.082
	MindCore	INT8	1649.395	**1.809**	**1.801**	**8.364**
ADFTD (EEG)	RTX A5000	FP32	**193.514**	71.749	13.284	58465.033
		INT8	432.155	72.269	13.284	58984.581
	Jetson Xavier AGX	FP32	503.351	4.232	2.789	1442.976
		INT8	1124.116	4.305	2.789	1516.386
	MindCore	INT8	3171.793	**1.812**	**1.801**	**11.701**
T-MNIST	**MindCore (SW)****	ternary	1857.955	1.806	1.782	**24.531**
	MindCore (HW)**	ternary	1053.019	1.832	1.801	31.218
	MindCore (POPC)**	ternary	**830.904**	**1.805**	**1.780**	26.075

* Operating frequency - RTX A5000: 1.695 GHz (boost), Jetson Xavier AGX: 1.36 Ghz, MindCore (FPGA): 100 MHz
** SW: software based accumulator, HW: SIMD based int16 accumulator, POPC: bitwise AND + popcount accumulator

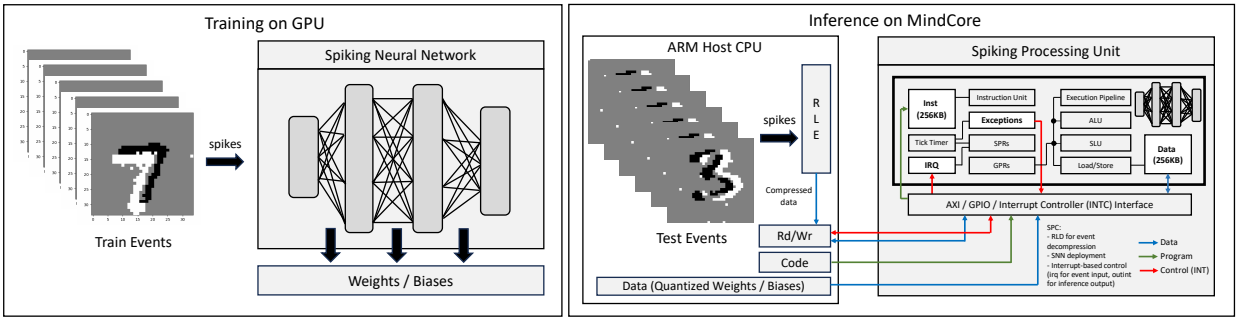


Fig. 3: Training and Inference setup for MindCore. The left block shows the training in full precision on GPUs. The right block shows the entire flow for inference. The ARM host sets up communication with MindCore, writes the generated program and data binaries to memory, sends samples over for inference, and calculates accuracies. MindCore itself receive samples of data and accumulates spike counts over time.

uses little-endian representation, whereas the RISC-based SPU uses big-endian representation.

Afterward, the ARM core resets the SPU to safely load the weight parameters, activation buffers, and input frames into memory. These data blocks are also written with byte-swapped ordering. Once initialization is complete, the ARM core signals the SPU via a GPIO2 interrupt. This causes the SPU to exit its idle state and begin computation.

Upon activation, the SPU sequentially loads the weights, activations, and inputs from memory, accumulates spike counts over time, and writes the resulting spike counts back to memory. When inference is complete, the SPU asserts an outgoing interrupt through the AXI INTC and returns to the idle state. The ARM core then catches this interrupt, writes the next input frame into memory, and issues another incoming interrupt. This completes one iteration of the bidirectional handshake cycle and only stops once entire dataset is processed.

This procedure remains consistent across models, except for N-MNIST, which requires decompression at every timestep. Like other SNNs, the N-MNIST model integrates spike activity over time, but its inputs are stored in a compressed RLE+VLQ format. Therefore, the system must decompress one timestep of input at a time before accumulation.

B. Verification and Debugging Infrastructure

During hardware–software co-design, an integrated verification framework was employed to ensure consistency between simulation and FPGA execution. A behavioral simulator written in Python reproduces instruction-level timing and spike propagation, serving as a golden reference for functional validation. The same testbench vectors were applied to the RTL simulation in Vivado to confirm bit-accurate results.

For system debugging, a lightweight trace monitor was embedded within the FPGA design, enabling non-intrusive capture of spike event flows and accumulator outputs through the Joint Test Action Group (JTAG) interface. These traces were used to confirm the correctness of SIMD and spike-vector instruction execution under varying workloads. The combined use of behavioral and hardware-in-the-loop verification reduced the development cycle and improved runtime reliability.

TABLE IX: Inference result

Dataset	Environment	Batch Size	Precision	Accuracy (%)
MNIST	CuPy	64	FP32	96.216
	CuPy	64	INT8	95.394
	GCC	1	INT8	94.080
	MindCore	1	INT8	95.000
N-MNIST	CuPy	64	FP32	95.100
	CuPy	64	INT8	94.860
	GCC	1	INT8	95.100
	MindCore	1	INT8	95.000
ADFTD (EEG)	CuPy	64	FP32	99.860
	CuPy	64	INT16	99.850
	CuPy	64	INT8	99.720
	GCC	1	INT8	99.720
	MindCore	1	INT8	100.000
T-MNIST	CuPy	64	ternary	83.405
	GCC	1	ternary	78.87
	Mindcore (SW)*	1	ternary	80.000
	Mindcore (HW)*	1	ternary	81.000
	Mindcore (POPC)*	1	ternary	80.000

* SW: software accumulator, HW: SIMD accumulator, POPC: popcount accumulator

C. System Performance and Comparative Analysis

Performance evaluation was conducted using identical workloads across three hardware targets: an RTX A5000 GPU, an NVIDIA Jetson Xavier embedded GPU, and the MindCore FPGA prototype. Tables VIII and IX show that, for event-based visual inference and biomedical classification tasks, MindCore demonstrated power consumption nearly two orders of magnitude lower than the embedded GPU, while maintaining comparable inference accuracy. MindCore's latency is as expected as a result of the considerably slower clock cycle. An examination of T-MNIST experimental results confirm that our spike-vector instructions reduced average computation latency by 45% for SIMD based instructions and 55% for popcount based instructions compared to equivalent multi-instruction sequences, validating the efficiency of the custom ISA.

V. Conclusion

This work presented MindCore, a spike-driven programmable accelerator designed for on-device neuromorphic computing. The architecture combines a RISC-based controller, custom spike-processing instructions, and efficient event compression to achieve real-time, low-power inference across diverse SNN models. Implemented and verified on a Xilinx ZCU104 FPGA, MindCore demonstrated significant power savings and scalability while maintaining functional flexibility. The experimental results from vision and biomedical benchmarks confirm its potential as a practical platform for embedded neuromorphic intelligence.

Acknowledgment

This work was supported in part by the Korea Electronics Technology Institute (KETI), which provided the Xilinx ZCU104 FPGA board used for experimental validation.

References

[1] Y. Zhang, P. Qu, Y. Ji *et al.*, "A system hierarchy for brain-inspired computing," *Nature*, vol. 586, pp. 378–384, 2020.

[2] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, pp. 607–617, 2019.

[3] A. Basu, L. Deng, C. Frenkel, and X. Zhang, "Spiking neural network integrated circuits: A review of trends and future directions," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, 2022, pp. 1–8.

[4] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106–122, Feb 2018.

[5] S. Moradi and G. Indiveri, "An event-based neural network architecture with an asynchronous programmable synaptic memory," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, 2014.

[6] S. G. Hu, G. C. Qiao, X. K. Liu, Y. H. Liu, C. M. Zhang, Y. Zuo, P. Zhou, Y. A. Liu, N. Ning, Q. Yu, and Y. Liu, "A co-designed neuromorphic chip with compact (17.9k f^2) and weak neuron number-dependent neuron/synapse modules," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 16, no. 6, pp. 1250–1260, 2022.

[7] D. A. Patterson and C. H. Sequin, "Risc i: A reduced instruction set VLSI computer," in *25 Years of the International Symposia on Computer Architecture (Selected Papers)*. New York, NY, USA: Association for Computing Machinery, 1998, pp. 216–230.

[8] M. M. Eljhani and V. Z. Kepuska, "Reduced instruction set computer design on FPGA," in *Proceedings of the 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering*, 2021, pp. 316–321.

[9] Y. Liao and D. B. Roberts, "A high-performance and low-power 32-bit multiply-accumulate unit with single-instruction-multiple-data feature," *IEEE Journal of Solid-State Circuits*, vol. 37, 2002.

[10] F. Akopyan, J. Sawada, A. Cassidy, and et al., "Truenorth: Design and tool flow of a 65 mw 1m-neuron programmable neurosynaptic chip," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1147–1164, 2015.

[11] M. Davies, N. Srinivasa, T.-H. Lin, and et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[12] Intel Neuromorphic Computing Lab, "A look at loihi 2," Online: Open Neuromorphic Computing Community, 2022.

[13] C. Pehle and et al., "The brainscales-2 accelerated neuromorphic system," *Frontiers in Neuroscience*, vol. 16, p. 795876, 2022.

[14] C. Mayr, S. Höppner, S. Furber, and et al., "Spinnaker 2: A 10 million core processor system for brain simulation and machine learning," *arXiv preprint*, 2019.

[15] D. Ma, X. Jin, S. Sun, and et al., "Darwin3: a large-scale neuromorphic chip with a novel isa and on-chip learning," *National Science Review*, vol. 11, no. 5, p. nwae102, 2024.

[16] BrainChip Holdings Ltd., "Akida neuromorphic processor," White Paper / Product Brief, 2023.

[17] P. Zhou, Q. Yu, M. Chen, Y. Wang, L. Meng, Y. Zuo, N. Ning, Y. Liu, S. Hu, and G. Qiao, "A 0.96 pj/sop, 30.23 k-neuron/mm 2 heterogeneous neuromorphic chip with fullerene-like interconnection topology for edge-ai computing," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2024, pp. 1–5.

[18] A. Birajdar, H. Agarwal, M. Bolia, and V. Gupte, "Image compression using run length encoding and its optimisation," in *Proceedings of the 2019 Global Conference for Advancement in Technology*, 2019, pp. 1–6.

[19] Y. Li and S.-J. Lin, "Removing redundancy in little-endian base 128 and the efficient decoding approach," *IEEE Communications Letters*, vol. 24, no. 11, pp. 2411–2415, 2020.

[20] A. Deb and D. Selvakumar, "Adaptive leaky-integrate-and-fire neuron model design and analysis," in *Proceedings of the 2024 IEEE 4th International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI SATA)*, 2024, pp. 1–6.

[21] M. Brehove, S. A. Tumpa, E. Kyubwa, N. Menon, and V. Narayanan, "Sigma-delta neural network conversion on loihi 2," 2025.

[22] S. B. Shrestha, J. Timcheck, P. Frady, L. Campos-Macias, and M. Davies, "Efficient video and audio processing with loihi 2," 2023.

[23] B. Ahn, "Implementation of a 12-million hodgkin-huxley neuron network on a single chip," 2020.

[24] E. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[25] M. D. Florio, A. Kahana, and G. E. Karniadakis, "Analysis of biologically plausible neuron models for regression with spiking neural networks," 2023.

[26] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[27] H. Wu and et al., "Integer quantization for deep learning inference: Principles and empirical evaluation," 2020.

[28] T. Gafni, A. Karnieli, and Y. Hanani, "Dual precision quantization for efficient and accurate deep neural networks inference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2025, pp. 3259–3269.

[29] R. Razani, G. Morin, and et al., "Adaptive binary-ternary quantization," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2021, pp. 4608–4613.

[30] P. Chen, B. Zhuang, and C. Shen, "Fatnn: Fast and accurate ternary neural networks," in *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5199–5208.

[31] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 2010, accessed: 2025-10-14.

[32] G. Orchard, G. K. Cohen, N. Jayawant, and N. Thakor, "N-mnist: Neuromorphic mnist dataset," https://www.garrickorchard.com/datasets/n-mnist, 2015, accessed: 2025-10-14.

[33] A. Miltiadous, K. D. Tzimourta, T. Afrantou, P. Ioannidis, N. Grigoriadis, D. G. Tsalikakis, P. Angelidis, M. G. Tsipouras, E. Glavas, N. Giannakeas *et al.*, "A dataset of scalp eeg recordings of alzheimer's disease, frontotemporal dementia and healthy subjects from routine eeg," *Data*, vol. 8, no. 6, p. 95, 2023.