Article | Received Day Mon Year; Accepted Day Mon Year; Published Day Mon Year
https://doi.org/10.55092/xxxx

Energy-efficient spiking neural network implementation for a retinal prosthesis

Marwan Besrour[1,2], Jacob Lavoie[1,2], Takwa Omrani[1,2], Jérémy Ménard[1,2], Esmaeil Ranjbar-Koleibi[1,2], Gabriel Martin-Hardy[1,2], Konin Koua[1,2], Mounir Boukadoum[3] and Réjean Fontaine[1,2]*

[1]Department of Electrical Engineering and Computer Science, Université de Sherbrooke, Sherbrooke, QC, Canada
[2]3IT Interdisciplinary Institute for Technological Innovation, Université de Sherbrooke, Sherbrooke, QC, Canada
[3]Département d'informatique, Université du Québec à Montréal, Montréal, QC, Canada

* Correspondence author; E-mail: Marwan.Besrour@Usherbrooke.ca.

Highlights:

- Highlight 1
- Highlight 2
- Highlight 3

Abstract: The quest for visual rehabilitation via retinal implant technology remains a complex, multi-dimensional endeavor. Retinal implants are biomedical devices surgically introduced into the ocular region. They propose a novel treatment for degenerative retinal pathologies. One of the most challenging aspects is replicating the retina's natural scene encoding. Many investigative teams have consistently pursued strategies to overcome this intricate challenge. Still, no one has effectively generated a device mirroring 20/20 vision. This paper presents a proof-of-concept framework that models the computational circuitry of the human retina using spiking neural networks (SNNs) and implements the model on a mixed-signal application-specific integrated circuit (ASIC) employing leaky integrate-and-fire (LIF) neurons. The proposed neuromorphic system on chip (NeuroSoC) demonstrates energy and area efficiency, achieving an energy consumption of approximately 25 pJ per synaptic operation and operating within an active silicon area of about 1 mm square. A detailed characterization of the chip is provided, including measurements of energy consumption per inference, an inference time of 1.264 ms, and validation over process, voltage, and temperature variations. Additionally, a Python-based emulation framework derived from chip measurements is developed and integrated with machine learning techniques, yielding a post-training quantization accuracy of 80.3% over 4-bit resolution on a synthetic retinal dataset modeled after Parasol ON retinal ganglion cells using the CIFAR-10 dataset. These quantitative performance metrics underscore the effectiveness and impact of our approach for potential retinal implant applications.

Keywords: neuromorphic circuit, spiking analog CMOS neuron, machine learning, spiking neural networks (SNN), edge ML, biomedical retinal implant, retinal ganglion cells

1. Introduction

The retina is a complex natural information processing system located at the back of the eye. A lens focuses photons passing through the pupil onto the sensory neuroepithelial layer of the retina with light projected on a smaller scale as an inverted image onto the 130 million photoreceptor cells (120 million rods and 6.5 million cones) in the outermost layer. Cones provide chromatic (color: red, green, and blue) images with high spatial resolution, while rods enable achromatic vision with lower spatial resolution under dim lighting conditions. The light patterns of the projected image are subsequently transformed into electrical and chemical signals. These signals activate a complex circuit of retinal neurons: horizontal, bipolar, amacrine, and ganglion cells [1]. The visual information from the 130 million photoreceptors is compressed into electrical and chemical signals for transmission by 1.2 million highly specialized neurons, whose axons form the optic nerve. The optic nerve conveys these signals via the lateral geniculate nucleus to the brain's primary visual cortex, where objects and scenes are reconstructed in four-dimensional space and time [1].

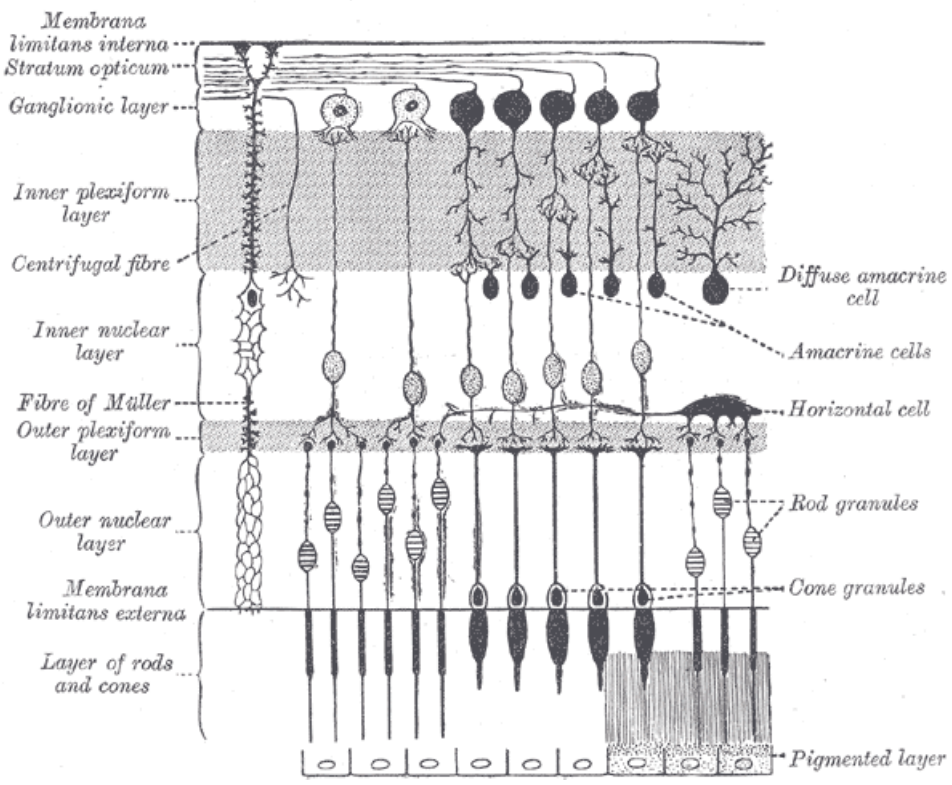

Figure 1. This figure illustrates the internal architecture of the retina, a three-layered structure of the central nervous system. The photoreceptors (rods and cones) reside in the outermost layer adjacent to the pigmented epithelium, where they convert incoming light into electrical signals. The middle layer contains bipolar cells, which relay these signals to the innermost layer composed of retinal ganglion cells. The axons of the ganglion cells converge to form the optic nerve, transmitting visual information to the brain. Horizontal and amacrine cells, also depicted in the figure, establish lateral connections within and between layers, refining, and modulating signal processing [2].

Figure 1 depicts the complex cellular organization of the retina. Different layers and types of cells contributing to vision can be observed. Starting from the outside, the first layer consists of pigmented epithelial cells. Next come the photoreceptors: the rods, which are responsible for vision in low light and are shown in gray, and the cones, which detect colors and are indicated in red, green, and blue. Together, they form the rods and cones layer. Beneath this lies the external limiting membrane, followed by bipolar cells, horizontal cells, and amacrine cells, all of which serve as intermediaries in the processing of the visual signal. Blood vessels supplying the retina with nutrients and oxygen are also represented. Finally, the retina includes the ganglion cells, whose axons form the optic nerve and transmit visual signals to the brain. This structure is supported by the internal limiting membrane at the base. The entire cellular organization is essential for converting optical images into electrical signals that the brain can interpret as visual images.

Retinal implants are biomedical devices surgically introduced into the ocular region. They generally encompass a data processing unit either integrated in the ocular region or as an external module. Retinal implants also require a power source, often implemented with batteries or wired connections [3–8].

Such devices employ a visiotopic stimulation approach, assuming that the retina acts like a digital camera and handles signals and stimulation configurations pixel-by-pixel at defined moments in time. At present, leading stimulation routines convert the digital input strictly into electrical impulses by reducing a fixed camera feed to the microelectrode array's dimensions [9]. Such procedures disregard that the RGC axons targeted for emulation are both cross-stimulated and subtype-specific, thus inevitably producing phosphene, which manifest as pseudo-persistent light bursts rather than truly effective vision in practice. Hence, individuals fitted with retinal implants employing this visiotopic principle fail to achieve a practical vision [10]. Observed *in vitro*, RGCs exhibit sparse activations across temporal and spatial domains resembling a retinal code. In this paper, we employ Convis [11], a Python package derived from the PRANAS (Platform for Retinal ANalysis And Simulation) tool [12], to reproduce the retinal code and instruct a spiking neural network to emulate it with minimal power and resource demands, and reliably operate within the implant.

Similar approaches have been proposed in the literature for retinal prosthesis design. For instance, Eckmiller (1997) [13] proposed a retina implant concept based on a real-time simulator comprising 256 individually tunable spatio-temporal filters. This method models the retina as an ensemble of independent filters, each with distinct spatial and temporal parameters, effectively capturing key aspects of primate retinal processing. Similarly, the IRIS2 system developed by Pixium Vision employs a neuromorphic image sensor to provide continuous, real-time responses to visual input, thereby efficiently encoding dynamic scenes through neuromorphic signal processing [14]. Building on these earlier innovations, our work trains an artificial spiking neural network and implements it on a neuromorphic CMOS chip for real-time applications.

Artificial neural networks can handle large datasets and approximate complex functions. More sophisticated machine learning, including spiking neural networks, could uncover hidden patterns and guide advanced retinal neurostimulation strategies. Advanced computing platforms have been essential to the current success of artificial intelligence (AI) research, driven by machine learning (ML) and its deep learning (DL) instantiation [15, 16].

Hence, there is a growing demand for the full implementation of DL models on the edge, using

resource-efficient hardware to develop. In this regard, neuromorphic computing is a practical and promising avenue to create faster and energy-efficient edge computing devices [17]. Performing real-time parallel data processing on the Edge is a big challenge due to the limited resources and power supply available, even with slimmed down GPUs for embedded applications such as NVIDIA's Jetson Nano [15, 16, 18]. One class of computing architectures partially took up the challenge by focusing on inference-only accelerator circuits like those found in the Coral Tensor Processing Unit (TPU) from Google [19] and the Neural Compute Stick from Intel [20]. Those proprietary integrated circuit designs can reach relatively low power consumption ($\leq 1Watt$) but do not afford customization. There exist also the event-driven data processing and non-Von Neumann architectures of neuromorphic processors such as Loihi [21] and Truenorth [22] which can significantly reduce the magnitude of the challenge, in addition to allowing on-chip learning. However, the latter feature significantly increases the on-chip communication between the neuromorphic cores, which increases both the circuit complexity and the power budget [17].

The retinal implant depicted in Figure 2 and Figure 3 marks the earliest prototype constructed at Université de Sherbrooke. It contains an external processing module, a wireless interface, and a stimulator linked to a multi-electrode array (MEA). Its external graphical processor converts incoming imagery into spiking signals and securely delivers both data and power consistently to the stimulation controller via an optical link[23]. In earlier designs, including ArgusII and AlphaIMS [3, 6], data and power cables traversed the eye wall, thus significantly raising the likelihood of much more serious postoperative complications [9, 24, 25]. Ultimately, the stimulator links to a multi-electrode array (MEA), which directly triggers action potentials or spikes in the RGCs by establishing an electrical field.

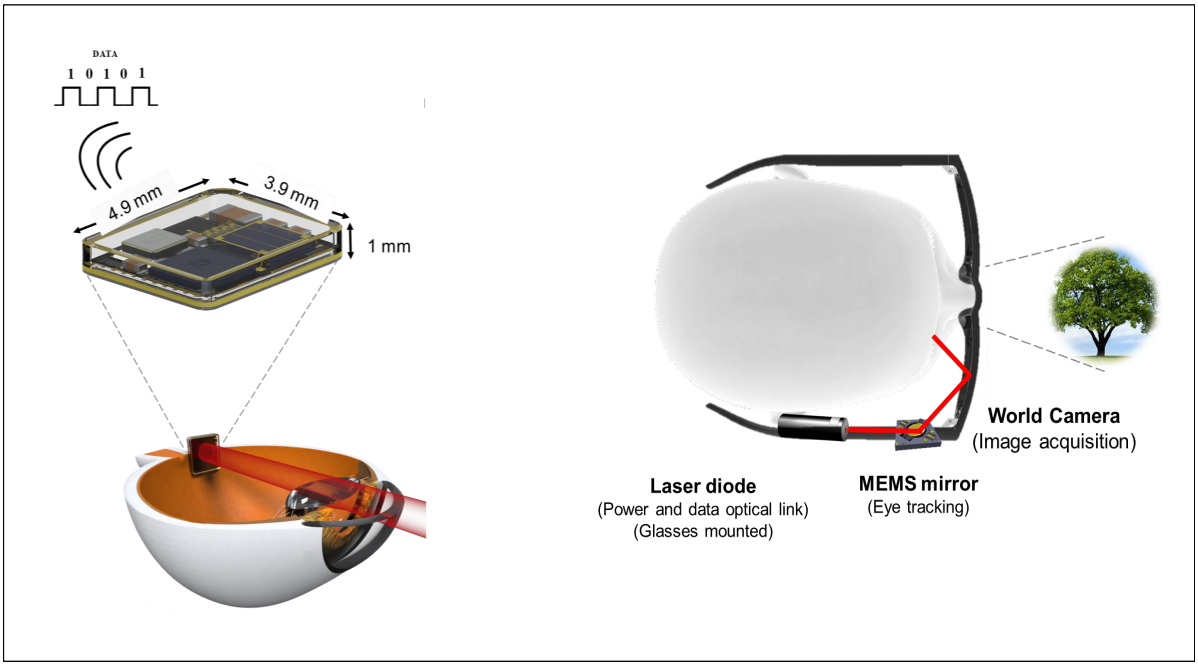

Figure 2. Sherbrooke Retinal Implant System Architecture. An 850-nanometer laser, modulated in intensity to deliver both power and data, is strategically directed toward an epiretinal implant via a controlled MEMS mirror. An external camera provides image acquisition to support visual feedback.

This paper assesses whether the Sherbrooke implant could be fully adapted by creating an SNN-based

intelligent controller as a graphical processing module and handling imagery on a dedicated ASIC. Achieving this objective necessitates observing stringent area and power limitations inherent to the implant while delivering accurate stimulation patterns to the retina-to-machine interface.

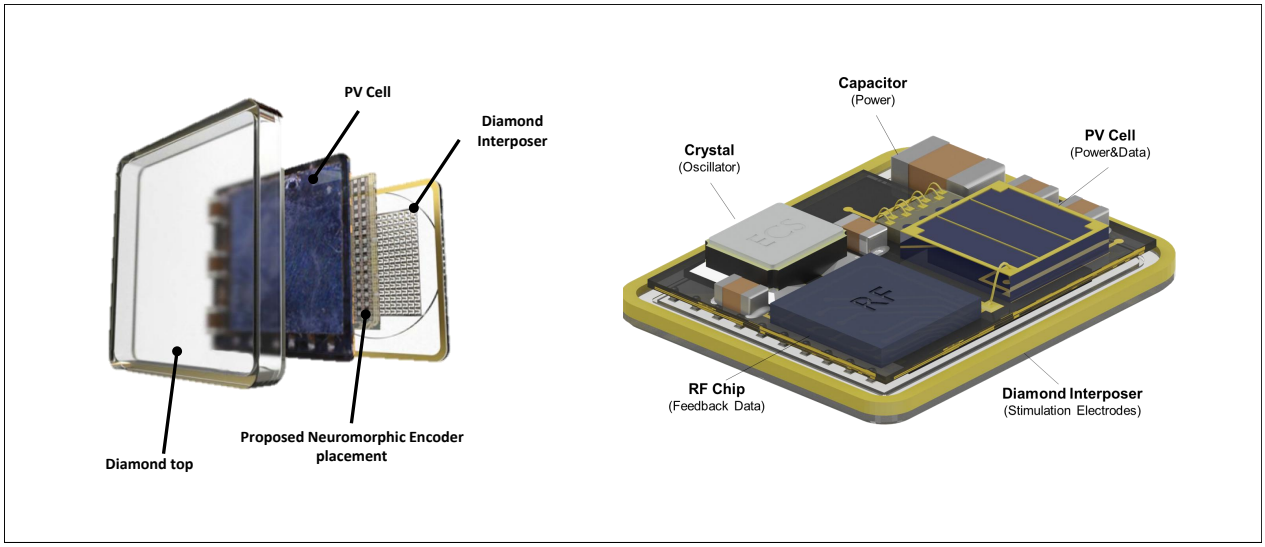

Figure 3. Retinal implant system architecture for power supply and data delivery. An 850-nanometer laser strategically directed toward the implant delivers power and data to the internal implant. This implant is equipped with a multi-junction photovoltaic cell that absorbs the incoming infrared light, providing power to the CMOS stimulator ASIC. Simultaneously, a photodiode decodes the data from the modulated laser beam to deliver the stimulation patterns through an array of microelectrodes [26].

2. Materials and methods

2.1. Materials

2.1.1. Retinal ganglion cells dataset

Figure 4 illustrates how the retinal ganglion cell dataset is generated using the Python library Convis, which employs GPU acceleration and is based on the backbone of PRANAS software. The retinal configuration includes both ON and OFF parasol and midget cells, and the resulting raster plots are subsequently post-processed to obtain vectorized average firing patterns of the retinal ganglion cells. In this workflow, each input from the CIFAR-10 dataset is converted to a monochrome image before being passed through Piranha to produce a raster plot. These plots are then transformed into average firing patterns for each simulated retinal neuron. Finally, both the input and the corresponding firing patterns are recorded as entries in the `Convis` dataset [11].

To address the challenge of reduced visual acuity in retinal implants and inform type-aware neurostimulation strategies, we implemented a procedure to generate controlled recordings of retinal ganglion cell (RGC) responses. This procedure utilized the PRANAS library's Convis module, a Python-based tool that simulates visual processing along various cell pathways. As a representative natural scene stimulus set, we employed CIFAR10, which contains a diverse range of object categories and spatial frequencies. Such diversity is crucial for evaluating the fidelity of implant-driven responses and ensuring that reconstructed visual fields are not limited to artificially constrained patterns.

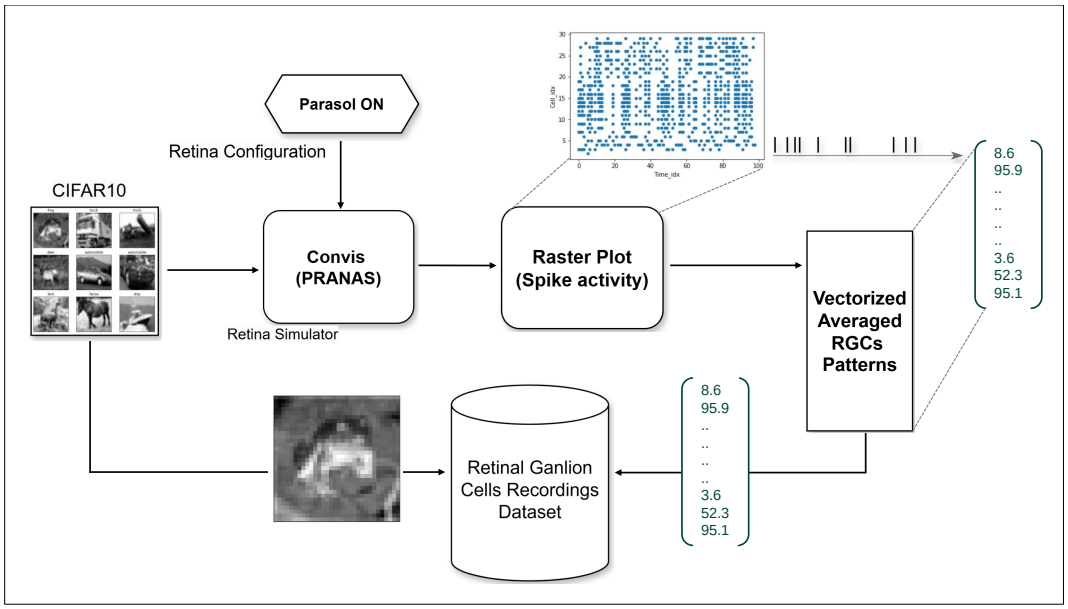

Figure 4. Retinal ganglion cell dataset generation using Convis and PRANAS: CIFAR-10 images are converted to monochrome, processed by Piranha to yield raster plots, and then transformed into vectorized average firing patterns.

In the simulation, a simplified model of the primate retina that includes only ON-type parvocellular (Midget) and magnocellular (Parasol) cells was configured. These RGC types were chosen due to the fact that they form 75% of all RGCs [27]. Each CIFAR10 image was presented to the modeled retina for a fixed period duration, e.g. 100 ms, allowing the temporal integration of luminance and contrast features. The output consisted of spike trains recorded at each RGC output. The spike trains for each RGC were aggregated during the exposure period into frequency-based vectors representing the average firing rates, enabling subsequent analysis and the development of a spiking neural network that replicates the inherent patterns of RGC activity. For example, if a given RGC spikes 2100 times during 100 ms, dividing the number of spikes by the exposure period yields an average spiking frequency of 21 kHz.

2.1.2. Neuromorphic chip design

Figure 5 shows the chip's internal architecture alongside a microphotograph. The diagram illustrates three fully connected layers with their synapses, labor shifters, and counters, as well as the placement of the digital controller and the SPI interface. The microphotograph features the pad ring and the seal ring, providing insight into the wire-bonding process that connects the chip to a DIP-20 package. This package is then placed on the PCB test bench, where a Raspberry Pi supplies input and simulation signals and reads the SPI data returned by the chip.

The neuromorphic chip uses the TSMC's 28 nm HPC process to ensure compact transistor dimensions and reduced leakage currents. It integrates a digital controller that manages the internal neural network operations and a Serial Peripheral Interface for configuration and data transfer. At the beginning of each inference, the neural network weights are transferred via the SPI interface from the Raspberry Pi (controller) to the SPI driver on the chip (peropheral), and then to the controlled register of the neural network. Similarly, the input weights are transferred in the same manner. The digital controller then triggers a start signal for a fixed duration to allow the spiking neural network to perform its computations,

with a spike counter at each neuron recording the number of activations. At the end of the computation, all spike counts are transferred to the digital controller, which sends the results to the SPI driver and back to the Raspberry Pi.

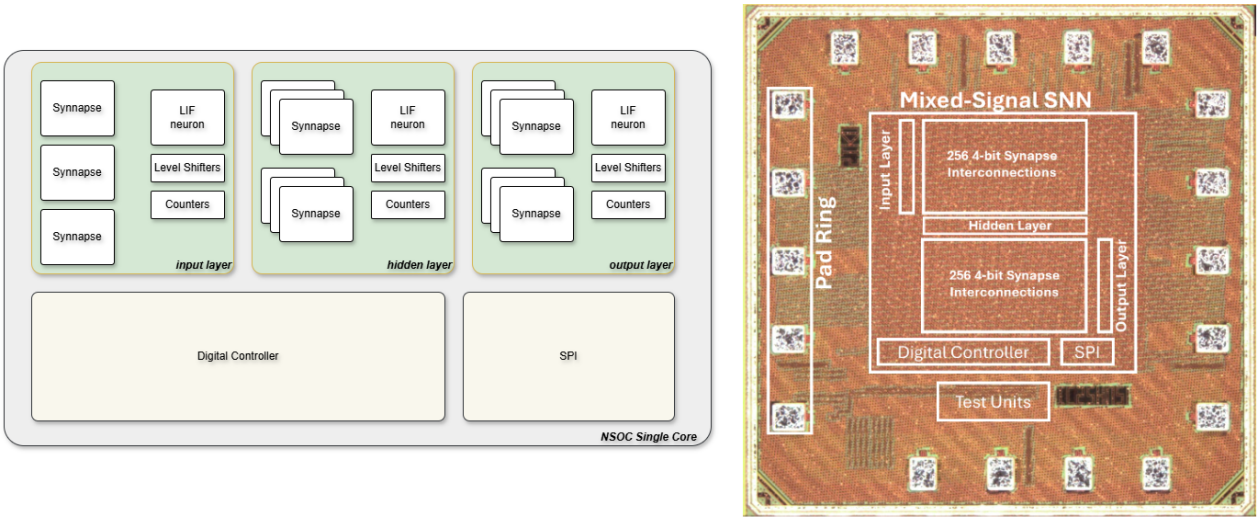

Figure 5. Chip Architecture, Packaging, and Test Bench Overview: This figure depicts the chip's internal architecture, featuring three fully connected layers with synapses, shift registers, and counters, along with the digital controller and SPI interface. It also illustrates the packaging process—with pad and seal rings for wire-bonding into a DIP-20 package—and the PCB test bench setup where a Raspberry Pi drives inputs and reads SPI data.

The device implements a fully connected spiking neural network with 48 neurons and 512 synapses arranged in three layers of 16 neurons each. The neurons follow a leaky integrate-and-fire model [28] that provides feedforward signal propagation without on-chip learning. The primary objective of this approach is to achieve energy efficiency, which is critical in applications such as retinal implants, where power must remain within stringent limits. The mixed-signal synapses use digital-to-analog converters to produce weighted current outputs in the sub-threshold region. These currents correspond to levels of approximately 12.5 pA, 25 pA, 50 pA, and 100 pA. Each neuron in the hidden layer receives input from 16 synapses, resulting in a maximum summed current of about 1.6 nA, a value compatible with the neuron's dynamic range. Operating synapses in sub-threshold mode reduces power consumption and supports low-voltage operation, an essential requirement for sustained implantation.

It should be noted that the network architecture implemented in this device is primarily a proof-of-concept to validate the feasibility of large-scale spiking neural network hardware implementations. The choice of 48 neurons and 512 synapses was dictated by the limited active silicon area (approximately $\simeq 1mm^2$) available on the chip and the constraints of our crossbar synapses matrix. Consequently, we implemented a fully connected, three-layer network with 16 neurons per layer to demonstrate energy-efficient spiking neural computation while operating within the stringent power and area limits required for implantable devices.

Figure 6 illustrates the finite state machine in the spiking neural network issues stop and start signals, defining the integration (or computation) time of the spiking neural network. After this period, all neurons transmit their data to the account register located near the digital controller. When the spike-ignorant

phase is active, competition concludes. The digital controller then transfers all count register data to the SPI peripheral, which returns the information to the Raspberry Pi.

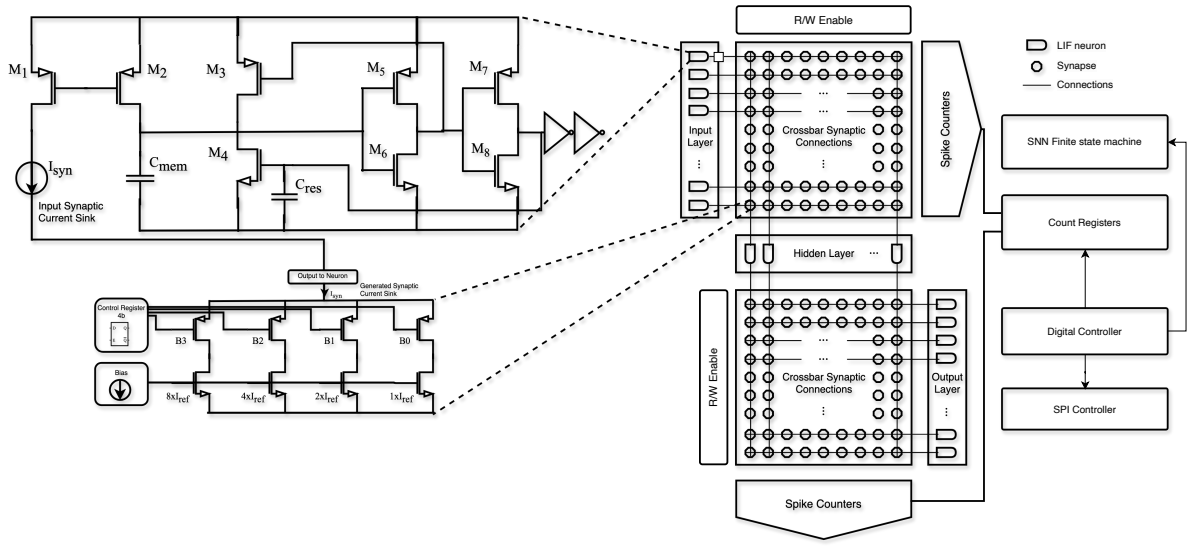

Figure 6. Finite state machine & data flow in spiking neural network: the SNN's finite state machine issues start and stop signals to define its integration period, during which neurons send data to an account register. Following a spike-ignorant phase, the digital controller transfers the data via the SPI peripheral back to the Raspberry Pi.

The fact that an on-chip learning circuit is non-existent lowers complexity and power consumption. Instead, network parameters can be calibrated offline, and the resulting synaptic weights are then configured digitally. The fully connected architecture ensures that each neuron can integrate signals from multiple sources, allowing for a certain level of representational flexibility despite the limited network size. The leaky integrate-and-fire model requires fewer transistors than more complex neuron models, further reducing area and static power.

To validate performance under realistic conditions, multiple chips were fabricated and mounted on custom-printed circuit boards. These boards facilitate controlled experiments to characterize current responses, neuron firing patterns, and overall power dissipation. The tests will indicate if we have a stable operation and will confirm that the selected architecture and circuit-level strategies meet the energy efficiency targets needed for the retinal prosthesis since the power budget for the latter is 35.6 milli-Watts.

Adopting a mixed-signal approach that employs digital-to-analog conversion at each synapse allows fine-grained control of synaptic weights. This configuration is compatible with the limited space and power budgets associated with retinal implants. The reduced complexity of feedforward signaling and the omission of on-chip learning minimize power overhead and silicon area. Consequently, this architecture can serve as a building block for larger-scale neuromorphic systems that require low power consumption, limited heat generation, and stable operation in environments where external power sources are constrained.

2.2. *Methods*

2.2.1. Neuromorphic chip characterization

Spiking Distribution of the Network Characterizing the neuromorphic chip involves quantifying its behavior across all neurons of each layer. The measurement testbench includes a custom printed circuit board (PCB), a source-measurement unit (SMU) the keithley 6784 picoammeter, and a RaspberryPI control system. The chip's Serial Peripheral Interface (SPI) port is connected to a logic analyzer and a RaspberryPI control system to ensure the precise configuration of synaptic weights, neuron parameters, and input stimuli Figure 7.

This setup enables a detailed examination of the network's internal computations. The characterization procedure begins by uploading a known set of synaptic weight values into the digital configuration registers integrated within the chip. Next, a defined set of input patterns is applied, corresponding to input spike trains with controlled timing and frequency. The resulting spiking distribution at each neuron's output node is captured within the internal counters and sent back using the SPI interface to the Raspberry PI controller. Successive verifications of the spiking distribution indicates that the integrated digital-to-analog converters (DACs) responsible for synaptic weight generation operate within the expected ranges. This characterization process establishes a baseline for understanding of the SNN spiking distribution.

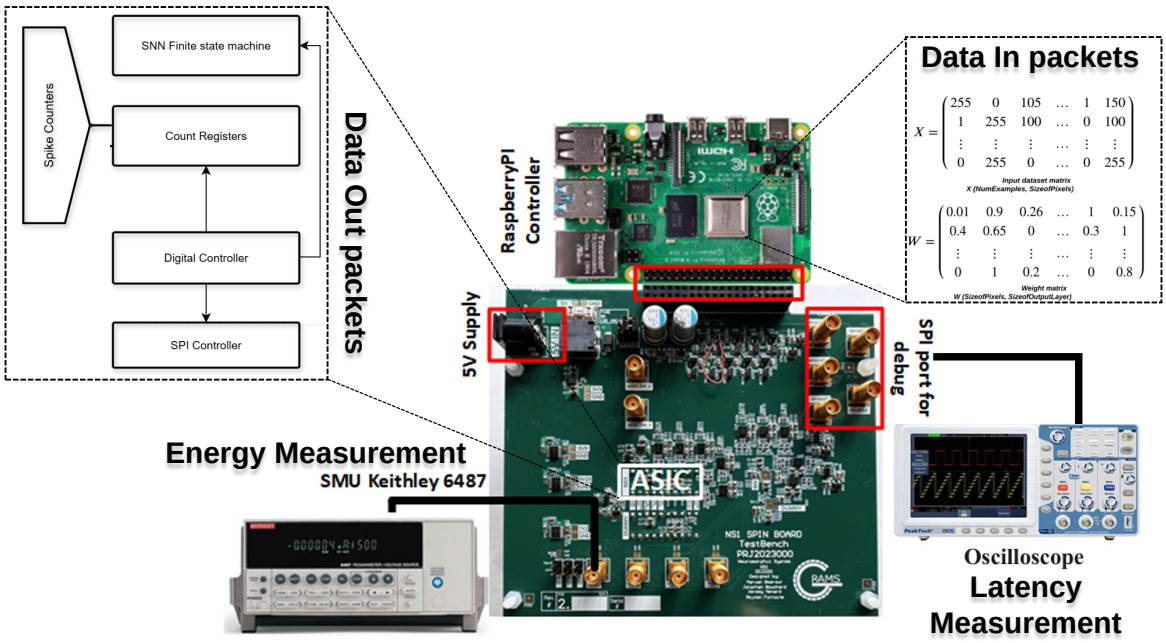

Figure 7. Experimental setup for chip characterization: A Raspberry Pi configures the network by sending and triggering computation via SPI while collecting spike counts from the chip.

Figure 7 illustrates the experimental setup used to characterize the chip. A Raspberry Pi controller, running a Python script with both network and input weights, sends the weights and the inputs to the ASIC through the SPI interface. Once configured, the internal finite state machine (FSM) issues a start signal to the on-chip digital controller, which initiates the neural network computation. After the integration period, each neuron's spike count is collected with the spike counters circuitry and saved to the Count

Registers as depicted in Figure 7. Afterward, the FSM stops the SNN computation, the digital controller sends the spike counts to the on-chip SPI peripheral interface which sends back to the Raspberry Pi the result of the neural network computation.

Energy Consumption Measurement Energy consumption per inference is quantified by monitoring the chip's dedicated power rail during the processing of predefined input patterns. A high-precision low-noise source-measurement unit (SMU), the Keithley 6784 picoammeter is connected to the supply port. Integrating this measured current over the inference duration, then multiplying by the supply voltage, provides the total energy consumed per inference. Prior to each measurement run, synaptic weight parameters are set to different dimensions allowing comparisons between different operating conditions. Various input patterns, spanning from sparse to dense spiking, are applied, and the resulting current profiles are captured.

Inference Time Measurement Measuring inference time involves applying known input spike patterns to the chip and recording how long the network requires to produce an output. A timing reference signal marks the start of inference, and the detection of output spikes at the final layer determines when a predefined steady-state firing pattern or expected output appears. Since the communication with the on-chip spiking neural network is established through the SPI interface, we have to take into account the read and write time of the SPI protocol. In addition, Multiple measurement runs are performed to minimize fluctuations and all data points are averaged to yield an overall timing figure. Comparing measured inference times against theoretical predictions validates the correlation with underlying neuromorphic chip design. In the case of the retinal prosthesis developed at Sherbrooke, the goal is to achieve a refresh rate of 30 frames per second, corresponding to an inference time of 33 milliseconds per frame. For energy consumption, as mentioned earlier, the value is approximately 35.6 milliwatts after conversion.

2.2.2. Hardware emulation model

A PyTorch-based framework was developed to emulate the chip's hardware-level behavior at a larger scale, ensuring that software-in-the-loop simulations reflect anticipated hardware performance. The environment encodes the chip's architectural constraints, including the 16-16-16 layer structure, quantized weights, and leaky integrate-and-fire neuron parameters. Surrogate gradient training approximates the Heaviside activation function during backpropagation, enabling gradient-based optimization while preserving spiking dynamics. A set of training datasets, including MNIST and Convis-CIFAR10, tests the network's capacity under diverse input conditions. After training with backpropagation and surrogate gradient techniques, weights are quantized to 4-bit resolution.

To convert image pixels into spikes, a rate coding technique is used. This method defines a fixed computation period for instance 100 milliseconds—divided into 1-millisecond steps, yielding 100 time slots for encoding each pixel. For example, a pixel with a value of 128 in an 8-bit resolution image (representing 50% intensity of the dynamic range from 0 to 255) is encoded by generating spikes during the first 50 time slots, with no spikes in the remaining slots. In contrast, a pixel with a value of 0 would produce no spikes, whereas a pixel with a value of 255 would generate spikes in all 100 time slots.

Converting MNIST images to 4-bit resolution ensures compatibility with the chip's synaptic weight

constraints. Each 8-bit pixel intensity (0–255) is mapped to a 4-bit value (0–15) by dividing by 16. This quantization aligns input data with hardware-level weight granularity and prevents values from exceeding the chip's dynamic range. Preliminary simulations in a PyTorch environment confirm that reducing pixel resolution to 4-bit does not severely degrade classification accuracy. Running inference with both full-precision and quantized inputs allows direct comparison of output performance metrics. After confirming acceptable results, the quantization step is integrated into the training pipeline. Training the network with quantized inputs ensures that the surrogate gradient optimization process adapts to the reduced-precision data. Multiple training runs confirms that the network converges reliably under these conditions.

Surrogate Gradient A surrogate gradient method addresses the training challenge posed by the non-differentiable Heaviside step function in spiking neurons. During forward passes, neurons produce binary spikes according to threshold conditions, preserving the desired spiking behavior. In backward passes, the absence of a defined derivative for the threshold operation impedes backpropagation gradient-based learning. To address this bottleneck, a differentiable surrogate is introduced. The softmax function is used in this case. During backpropagation, the surrogate gradient approximation substitutes the non-differentiable step, permitting standard optimizers to compute the gradients over non-differentiable function and update synaptic weights. By periodically monitoring loss and validation accuracy, it is possible to confirm that learning proceeds as expected. After convergence, the forward pass operates solely with the extracted charcterization from the LIF neuron, confirming that the trained model functions effectively under hardware-compatible spiking conditions.

Weights Quantization After the training phase, a weight scaling operation is performed which ensures that the learned synaptic weights fit the chip's hardware constraints. Although the network was trained with quantized inputs and surrogate gradient methods, the trained weights are in half-precision (16-bit) and are not aligned with the 4-bit resolution on-chip synapses. To address this aspect, a systematic scaling approach maps the trained weights onto discrete values. The process begins by extracting the trained weights and identifying their maximum and minimum absolute values. The distribution is then examined to determine the mean and maximum values, which guide the selection of an appropriate scaling factor. Finally, each weight is mapped to a four-bit representation, with zero assigned to zero and all remaining values aligned to the corresponding discrete levels.

The process begins by extracting the trained weight matrix and examining its distribution. The minimum and maximum values are used to identify an appropriate scaling factor in order to map the trained weight distribution to 4-bit resolution and maintain the overall distribution. The scaling factor is then applied, and the resulting values are rounded to the nearest 4-bit integer, ensuring that no weight exceeds the hardware's maximum allowable value. Subsequent verification steps involve running inferences on the model using these scaled weights.

A uniform symmetric quantization technique was selected where $w \in [\alpha, \beta]$ are the trained weights, and b is the chosen bitwidth where $b = 4$. Uniform quantization maps w to Q within

$$Q \in [-2^{b-1}+1, 2^{b-1}-1] \quad . \tag{1}$$

Where:

$$S = \frac{max(\mid \beta \mid, \mid \alpha \mid)}{2^b - 1}, \tag{2}$$

Which yields:

$$Q = \text{Clip}\big(\text{round}(w/S), -2^{b-1} + 1, 2^{b-1} - 1\big) \tag{3}$$

And finally the quantized weight as follows:

$$\hat{w} = SQ \tag{4}$$

3. Results

3.1. Hardware characterization results

Spiking Distribution of the Network Evaluating the network's spiking distribution involves presenting known input patterns and analyzing the spike counts in each layer. Input patterns are converted into spike trains that feed the input layer, where neurons emit spikes proportional to their assigned pixel intensities, which have been quantized to a 4-bit resolution. Internal counters and digital circuits monitor and record the activity of every neuron in the network over a predefined time window. Incremental variations in input frequency or pixel intensity produce predictable shifts in firing rates, demonstrating the network's adaptive behavior while preserving class-specific output patterns. The firing frequency of the implemented LIF neuron for different input synaptic current amplitudes increases with the input synaptic current. The dynamic range starts from 10 kHz up to 350 kHz in the 10 pA to 3000 pA input range, respectively.

Nevertheless, process, voltage, and temperature (PVT) variations introduce deviations in the spiking distribution. These discrepancies are mitigated through a post-training recalibration process using a software-based correction technique to counteract the variations. At the output layer, distinct spiking patterns correspond to different input classes, illustrating the network's ability to classify digit categories based on spike distributions. Stability is validated by repeating identical inputs and observing consistent spiking distributions across multiple trials and chip samples.

The spiking neural network (SNN) implemented on this chip contains three fully connected layers, each containing 16 neurons. Each synapse in the circuit is realized as a 4-bit-controlled digital-to-analog converter for current sources. When the synaptic weights are varied and the spiking distributions of all 48 neurons are examined, slight offsets caused by PVT variations are observed. For instance, neuron one in the first layer may fire at 11.5 kHz for an input synaptic weight of 2, while neuron five in the same layer fires at 25 kHz. To address these offsets and achieve uniformity across the network, a calibration method is employed. Following training, the SNN weights are quantized and transferred to the chip via the SPI interface. Before this transfer, the network's spiking distribution is evaluated on-chip to identify offsets, and a scaling factor is applied to the weights to compensate for the variations. This correction ensures that the network operates consistently, regardless of the inherent PVT variations.

Energy Consumption Measurements Energy consumption was measured by monitoring the supply current during inference. The instantaneous current over the entire inference period—from input spike arrival to output stabilization was recorded. Multiplying the integrated current by the supply voltage yields the total energy per inference. The chip's mixed-signal design, featuring sub-threshold analog synapses and

leaky integrate-and-fire neurons, achieved energy consumption on the order of a picojoules per inference. A measured average of 66.678 pJ per inference corresponds to roughly 0.126 pJ (0.25 V) per synaptic operation, considering half of the synapses are active and at their full capacity. Inputs eliciting higher spike activity increased energy usage slightly, reflecting the proportion of active synapses and neurons. Nonetheless, this increase remained within predicted range limits. This measured energy figure however, only represent the three-layered of 16 neurons each fully connected network. The former figure will be used to estimate the energy consumption of the large-scale emulation of the SNN.

To measure the energy consumption of a three-layer, 16-neuron fully connected network, the root mean square energy was recorded and divided by the total number of synapses. In this network, the number of synapses is calculated as 16 neurons by 16 neurons times two layers plus 16 input synapses which is equal to 528 synapses, yielding approximately 25 picojoules per synaptic operation. To estimate the energy consumption of a larger-scale neural network, the total number of synapses is computed and multiplied by 25 picojoules per synaptic operation. To provide a more detailed breakdown of power consumption across individual components, we measured the contribution of a single leaky integrate-and-fire neuron to be around 2 fJ per spike on average across its dynamic range [29] . Although this detailed analysis is discussed outside the scope of this paper, these measurements allow us to estimate that the contribution of only the neurons to the overall power budget is approximately 483 pJ per inference for a 400-128-10 neural network. This approach provides an acceptable estimate of the power consumption for a larger network implemented with the same architecture proposed in this paper.

Inference Time Measurements Inference time was measured by timing the interval between applying an input spike pattern and reaching a stable output. Under test conditions, the chip reached stable output firing patterns in 1.264 ms. This speed arises from the fully parallel, network architecture, where all neurons update their states simultaneously rather than in sequence. Analog synaptic circuits further reduce latency through charge-based computations. Varying input parameters, such as spike rates or weight configurations, did not significantly alter the inference time. The network consistently reached equilibrium within the same timeframe. In visual prosthesis applications, microsecond-scale inference times enable real-time processing of incoming sensory signals. This capability is essential in closed-loop environments requiring continuous adaptation. By achieving such low-latency performance, the chip paves the way for a range of neuromorphic real-time computing tasks.

3.2. Hardware model validation on the retinal dataset

A Python-based emulation model of the chip's hardware is implemented in PyTorch to provide a controlled environment for large-scale emulation of the fabricated SNN. This software-based environment replicates the architectural parameters, including the leaky integrate-and-fire neuron and spiking distribution derived from the characterization process. The 4-bit quantization scheme, central to the chip's synaptic weight format, is also integrated into the emulation. An SNN is trained to replicate the firing patterns of the RGC Parasol ON cells, leveraging surrogate gradient methods to optimize weights within the 4-bit quantization constraint. This hardware emulation enables testing under a range of input conditions that mirror those presented to the actual chip. During the test phase, feedforward activation functions are emulated using distributions obtained from previous spiking measurements. Additionally, Figure 8

illustrates randomly selected monochrome images from Convis-CIFAR10 used as input stimuli and their reconstructed counterparts using the trained SNN.

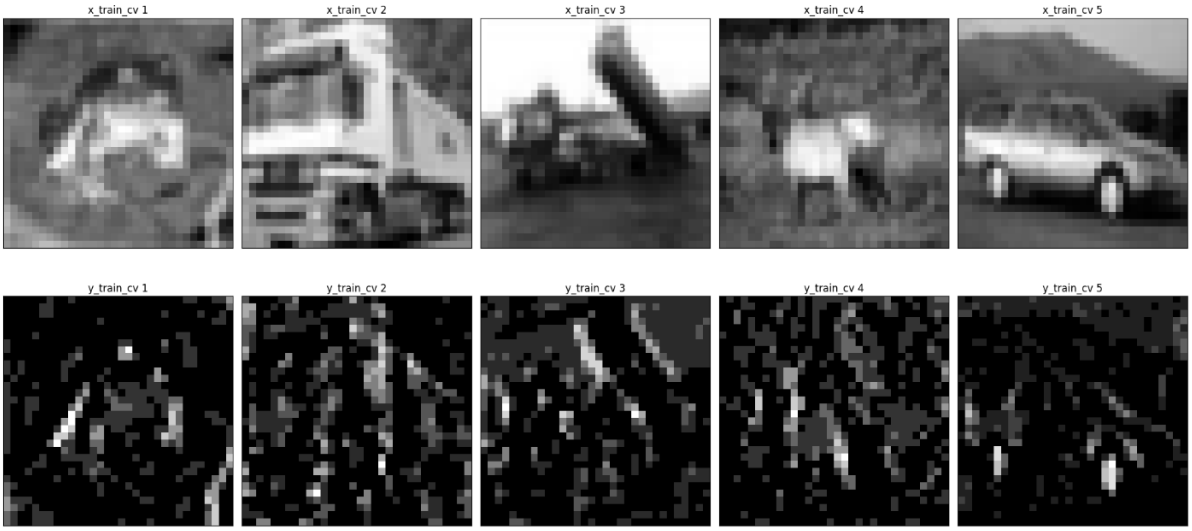

Figure 8. This figure presents selected images from the CIFAR10 dataset and their reconstructed counterparts obtained by the trained SNN using the architecture model [1024-512-256].

To demonstrate that the neural network faithfully reproduces the spikes corresponding to the natural activations of retinal ganglion cells (RGC), a computation period is defined—for example, 1 second (1000 milliseconds) with a computation step of 1 millisecond. During this period, all inputs (pixels) are converted into spikes, and the network performs its inferences. At the output layer, the neurons modeling the natural activations of the RGCs are examined. The observed activation patterns are averaged to facilitate analysis and then compared to the database generated with PRANAS. For each image, it is verified that the average vector of spiking frequencies from the output layer corresponds to the one used to train the network.

The reconstruction of the output spikes from the spiking neural network follows a process similar to converting pixels to spikes. A fixed computation period and step size are used to compute the average spiking frequency of each output neuron. For instance, when the output comprises 256 neurons, these values are arranged to form an image. In our approach, a 256-neuron vector is used to generate an image, which is then upscaled to 32×32 pixels to match the resolution of the CIFAR-10 dataset (1024 pixels). A custom function is applied to interpolate additional pixels, thereby scaling the image from 16×16 to 32×32 pixels using the CV" python library. To obtain an 8-bit pixel value from the average spiking frequency of each neuron, the frequencies are normalized by dividing by the maximum frequency observed for each inference, and then multiplied by 2^8, resulting in values between 0 and 255. Finally, the 256 values are reshaped into a 16 by 16 matrix, and the image is plotted. This approach provides a visual assessment of the network's ability to process natural scenes and produce interpretable outputs under hardware constraints.

Figure 8 presents randomly selected monochrome images from the Convis-CIFAR10 dataset and their

reconstructed counterparts obtained by the trained SNN using the architecture model [1024-512-256]. Table 1 presents the accuracy results of the trained SNN over the Convis-CIFAR10 dataset and reports the hyperparameters used in the training process. Although the accuracy metric of the proposed SNN does not compare to the state-of-art over CIFAR10, the result demonstrates a proof-of-concept that RGC behavior can be modeled and reproduced to a certain degree using CMOS implementation of analog spiking neural network.

Table 1. Summarizes the classification metrics of the trained SNN over the Convis-CIFAR10 dataset and reports the hyperparameters used in the training process.

Model Architecture	[1024-512-256]
Learning rate	0.0001
Hidden layer(s)	512
Batch Size	256
Epochs	100
Trained	(64-bit)
Quantized	(4-bit)
Test accuracy	84.5%
PTQ accuracy	**80.3%**

4. Discussion

Numerous neuromorphic systems have been proposed to achieve efficient spike-based computation, many focusing on large-scale digital implementations or mixed-signal devices with varying degrees of complexity. Compared to these existing efforts, the proposed chip demonstrates a unique combination of low-latency operation and high energy efficiency in a compact, fully connected network architecture. The proposed chip leverages sub-threshold analog synapses and low-complexity leaky integrate-and-fire neurons, achieving a reported SOP energy consumption on the order of 25 picojoules. This power budget is competitive with other known implementations while still delivering real-time inference speeds and accuracy. While the characterization and performance metrics obtained for the chip indicate stable operation, low energy consumption, and rapid inference, the limited scale of the implemented network constrains its immediate applicability to large-scale problems. With a total of 48 neurons and 512 synapses arranged in a modest 16-16-16 structure, the current chip does not approach the neuron counts found in state-of-the-art large-scale neuromorphic systems or biological neural networks. This limitation in neurosynaptic density means that the tasks that can be performed on the chip are restricted in complexity and representativeness.

In practical terms, to handle large and more intricate datasets or to model extensive sensory processing pathways, a substantially higher number of neurons and synapses would be necessary. Such scaling would likely introduce new design challenges, including increased on-chip communication overhead, more complex routing of signals between neurons and synapses, and the need for efficient memory storage solutions. Additionally, with larger networks, ensuring that energy efficiency and inference speed remain at the desired levels would become more challenging. More neurons and synapses would inherently increase the total number of operations per inference, potentially raising energy consumption unless the architecture or the underlying circuits are optimized further. The current size limitation, while suitable for demonstrating core principles and validating the integration of analog synapses with digital control

logic, prevents the direct application of the chip to large-scale tasks such as robust image recognition in complex environments or full retina-scale simulation. To advance beyond this stage, future designs must incorporate scalable architectures, hierarchical connectivity patterns, and potentially on-chip learning mechanisms or more efficient off-chip data-transfer protocols. In interpreting the results, it is important to recognize that the favorable energy and latency metrics represent conditions for a relatively small network. Scaling up would require careful consideration of circuit complexity, heat dissipation, packaging constraints, and fabrication costs. Thus, while the chip provides a valuable proof of concept, it remains limited in terms of neurosynaptic density and consequently cannot yet serve as a direct replacement for large, biologically realistic networks or high-performance machine learning models requiring extensive parallel processing.

The validation of the Python-based emulation model through direct comparison with the hardware results demonstrates that software-based approximations can effectively predict the chip's behavior. By incorporating the same neuron models, synaptic quantization levels, and input encoding strategies, the emulation framework closely replicates the hardware's response to given stimuli. This correlation between measured hardware performance and software predictions confirms that the surrogate gradient training, weight quantization procedures, and pixel-to-spike conversion methods are compatible with the chip's operational constraints. The validated scaling approach ensures that complex training algorithms performed in software can translate into hardware-executable weight configurations. The accuracy of the retinal code replication, as evaluated in this study, must be interpreted with caution. Although the chip-based SNN produced distinct and stable spiking patterns that resembles the expected output of a retinal ganglion cell layer, these results relied on input data generated through software simulations rather than recorded in-vivo measurements. Without a direct comparison to biological recordings obtained from living retinas, it remains uncertain how closely the implemented SNN and its firing patterns align with actual retinal activity under natural viewing conditions.

In addition, it is critical to achieve one-to-one stimulation between the electrode array and retinal ganglion cells to yield selective targeting. However, addressing these technical advances is beyond the scope of the present study. This assumption is predicated on the eventual realization of one-to-one stimulation, whereby each electrode precisely targets a single retinal cell type. In the current work, such precision is not claimed. Instead, the limitation of high-acuity implant efficiency is emphasized in the absence of targeted stimulation. Moreover, sophisticated signal conditioning is necessary to ensure that the electrical stimulus can be finely tuned to the specific response characteristics of individual retinal cells. A deeper understanding of the spatial arrangement and functional diversity of retinal ganglion cell mosaics is also critical, as this knowledge would enable the mapping of visual input to the appropriate cell type with high accuracy. In this context, the discussion highlights that while the current approach offers promising steps forward, it also underscores the limitations imposed by existing stimulation technologies. Future research must therefore focus on developing strategies to identify and selectively stimulate individual cell types, which is essential for the successful translation of these technologies into effective, high-acuity retinal implants.

Despite these limitations, the demonstrated ability to produce stable spike patterns that mimic aspects of the retinal code suggests that, if combined with more realistic stimulation protocols and electrode technologies, neuromorphic devices could lead to improved visual acuity in retinal implants. The key

would be refining the network configuration, input encoding methods, and stimulation strategies to approach the complexity and richness of biological signals. Future research should involve integrating data from in-vivo experiments, refining the spatial-temporal properties of the SNN's input signals, and exploring adaptive stimulation techniques that dynamically adjust to the patient's feedback. While the current results are promising, translating these findings into clinical improvements in visual acuity will require addressing the assumptions made, validating against biological data, and developing corresponding hardware interfaces capable of delivering precisely targeted stimuli. Only through such iterative refinement and close collaboration with biomedical research can the potential of neuromorphic retinal implants be fully realized.

Table 2. Chip Characteristics and Measured Performances Compared With the State of the Art of Spiking Neuromorphic Circuits.

Author	This work	Modha	Orchard	Stujit	Frenkel	Frenkel	Moradi	Akopyan
Publication	ELS NanoAI 2025	Science 2023	IEEE WSPS 2021	Frontiers 2021	ISCAS 2020	IEEE TBCAS 2019	TBioCAS, 2017	TCAD, 2015
Chip name	—	IBM NorthPole	Loihi 2	UBRAIN	SPOON	ODIN	DYNAPS	LOIHI
Implementation	Analog	Digital	Digital	Digital	Digital	Digital	Mixed-signal	Digital
Technology	28nm	12nm	14nm	40nm	28nm FD-SOI	28nm FD-SOI	0.18µm	28nm
Neurosynaptic core area [mm²]	0.24	2.52	0.41	1.42	0.26	20	7.5	11
# Izhikevich behaviors	1	NA	20	NA	NA	20	20	4
# neurons per core	48	16k	1M	NA	148	256	256	256
Synaptic weight storage	4-bit (DAC)	8/4/2-bit (RAM)	8/32-bit	4-bit	8-bit	(3+1)-bit (SRAM)	12-bit (CAM)	1- to 9-bit (SRAM)
Embedded online learning	No	No	Yes	No	Stochastic DRTP	SDSP	No	No
Synapses per core	528	4M	120M	20k	2.4k	64k	16k	64k
Time constant	Accelerated	Accelerated	Accelerated	Accelerated	Bio. to accel.	Bio. to accel.	Biological	Bio. to accel.
Neuron core density [neuron/mm²]	96	6.31k	2.14k	336	148	3.0k	34	2.56k
Synapse core density [syn/mm²]	1056	1.6M	2.1M	20k	2.4k	8.4k	2.1k	2.5M to 256k
Supply voltage	0.2V-0.4V	0.9V-1.0V	0.5V-1.25V	1.1V	0.6V-1.0V	0.55V-1.0V	1.3V-1.8V	0.55V-1.0V
Energy per SOP	25 pJ	15 nJ	23pJ	308nJ	300nJ	12.7pJ	30pJ	26pJ

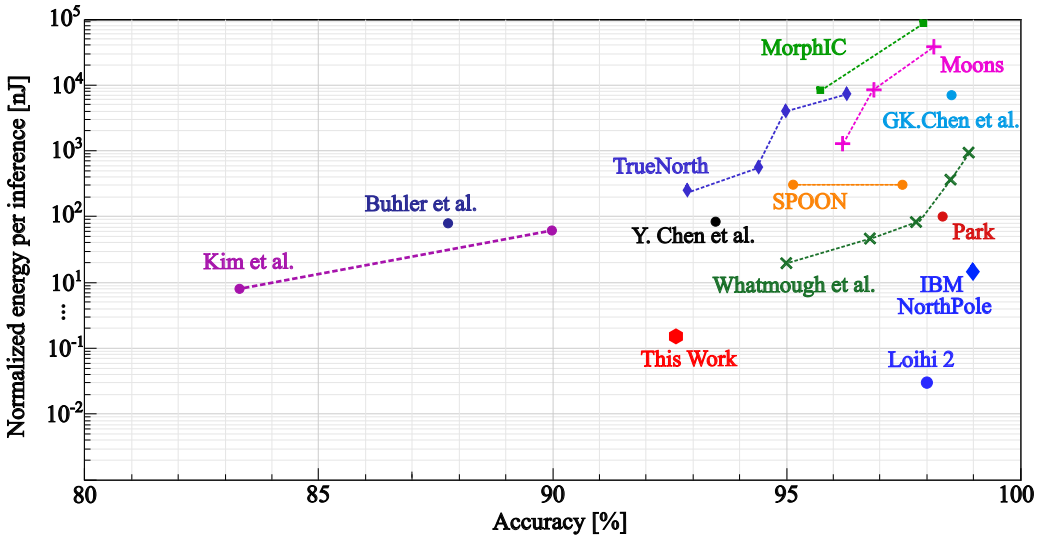

Figure 9. The figure illustrates the energy metrics of various reported neuromorphic platforms and their accuracy in classifying the MNIST dataset. The energy per inference is estimated from experimental measurements on a fully connected SNN [400-256-10], which was trained to classify MNIST with 92.7% accuracy, resulting in an energy per inference of 1.65 nanojoules for this configuration.

5. Conclusion

This paper demonstrates the realization of a fully connected spiking neural network (SNN) implemented on a mixed-signal application-specific integrated circuit (ASIC) integrating sub-threshold analog

synapses, leaky integrate-and-fire neurons, and a digital controller. The chip demonstrates that compact neuromorphic hardware can scale up to execute higher computational complexity neural networks on edge. The development of a Python-based emulation framework in PyTorch is a critical bridge between high-level algorithmic design and low-level hardware implementation. By reproducing the chip's network configuration, quantization levels, and neuron parameters in software, we ensured that training and optimization methods, such as surrogate gradient-based learning and weight scaling, translate seamlessly to the hardware domain. This alignment corroborates that software-optimized networks can be mapped and executed onto neuromorphic chips with minimal loss in accuracy or efficiency. By approximating aspects of the retinal code, the proposed chip offers insight into how neuromorphic hardware might deliver more biologically realistic stimulation patterns. Although further work is necessary to incorporate in-vivo data, refine electrode interfaces, and address the complexity of actual retinal connectivity, the preliminary results indicate that neuromorphic devices hold promise for improving visual acuity in patients with degenerative retinal diseases. This work establishes a proof of concept for low-power, low-latency neuromorphic implementation of an artificial spiking retina in CMOS destined to be implemented as a stimulation pattern generator in a second-generation retinal implant.

Acknowledgments

Conflicts of Interests

The authors declare no conflict of interest.

Authors' contribution

References

[1] CambridgePress. Experts Reviews in Molecular Medecine Cellular Organisation of the Retina. *Cambridge University Press* 2014 .

[2] Gray H, Carter HV. *Anatomy of the Human Body*, Philadelphia: Lea & Febiger1918. See "Book" section below. Bartleby.com: Gray's Anatomy, Plate 882.

[3] Cruz Ld, Coley BF, Dorn J, Merlini F, Filley E, *et al.* The Argus II epiretinal prosthesis system allows letter and word reading and long-term function in patients with profound vision loss. *Br. J. Ophthalmol* 2013, 97(5):632–636.

[4] Finn AP, Grewal DS, Vajzovic L. Argus II retinal prosthesis system: a review of patient selection criteria, surgical considerations, and post-operative outcomes. *Clin Ophthalmol* 2018, 12:1089–1097.

[5] Hahn P, Fine HF. Practical concepts with the Argus II retinal prosthesis. *Ophthalmic Surg. Lasers Imaging Retina* 2018, 49(10):742–746.

[6] Zrenner E, Bartz-Schmidt KU, Benav H, Besch D, Bruckmann A, *et al.* Subretinal electronic chips allow blind patients to read letters and combine them to words. *Proceedings of the Royal Society B: Biological Sciences* 2011, 278(1711):1489–1497.

[7] Edwards TL, Cottriall CL, Xue K, Simunovic MP, Ramsden JD, *et al.* Assessment of the Electronic Retinal Implant Alpha AMS in restoring vision to blind patients with end-stage retinitis pigmentosa. *Ophthalmology* 2018, 125(3):432–443.

[8] Stingl K, Bartz-Schmidt KU, Besch D, Braun A, Bruckmann A, *et al.* Artificial vision with wirelessly powered subretinal electronic implant alpha-IMS. *Proceedings of the Royal Society B: Biological Sciences* 2013, 280(1757):20130077.

[9] Olmos de Koo LC, Gregori NZ. The Argus II Retinal Prosthesis: a comprehensive review. *International Ophthalmology Clinics* 2016 56(4):39–46.

[10] Ayton LN, Barnes N, Dagnelie G, Fujikado T, Goetz G, *et al.* An update on retinal prostheses. *Clin. Neurophysiol.* 2020, 131(6):1383–1398.

[11] Huth J, Masquelier T, Arleo A. Convis: a toolbox to fit and simulate filter-based models of early visual processing. *Front. Neuroinf.* 2018, 12:9.

[12] Cessac B, Kornprobst P, Kraria S, Nasser H, Pamplona D, *et al.* PRANAS: a new platform for retinal analysis and simulation. *Front. Neuroinf.* 2017, 11.

[13] Eckmiller R. Learning retina implants with epiretinal contacts. *Ophthalmic Research* 1997, 29(5):281–289.

[14] Bloch E, Luo Y, da Cruz L. Advances in retinal prosthesis systems. *Therapeutic Advances in Ophthalmology* 2019, 11:1–16.

[15] Blouw P, Choo X, Hunsberger E, Eliasmith C. Benchmarking keyword spotting efficiency on neuromorphic hardware. In *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*, New York, USA, March 26–28, 2019, pp. 1–8.

[16] Nguyen AT, Drealan MW, Luu DK, Jiang M, Xu J, *et al.* A portable, self-contained neuroprosthetic hand with deep learning-based finger control. *arXiv* 2021, arXiv:2103.13452.

[17] Davies M, Wild A, Orchard G, Sandamirskaya Y, Guerra GAF, *et al.* Advancing neuromorphic computing with Loihi: a survey of results and outlook. *Proc. IEEE* 2021, 109(5):911–934.

[18] Cataldi G, Mercorelli P, di Mare L, Pascali M, Russo G, *et al.* Artificial intelligence for environment monitoring: a case study with deep learning and UAV images. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. 2020, pp. 264–268.

[19] Coral. Coral AI Accelerator Product Information, 2024.

[20] Intel. Intel Compute Stick STCK1A32WFC Specifications, 2024.

[21] Davies M, Srinivasa N, Lin TH, Chinya G, Cao Y, *et al.* Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 2018, 38(1):82–99.

[22] DeBole MV, Taba B, Amir A, Akopyan F, Andreopoulos A, *et al.* TrueNorth: accelerating from zero to 64 million neurons in 10 years. *Computer* 2019, 52(5):20–29.

[23] Lemaire W, Benhouria M, Koua K, Besrour M, Gauthier LP, *et al.* Retinal stimulator ASIC architecture based on a joint power and data optical link. *IEEE J. Solid-State Circuits* 2021, 56(7):2158–2170.

[24] Luo YHL, da Cruz L. The Argus® II Retinal Prosthesis System. *Progress in Retinal and Eye Research* 2016 50:89–107. 10.1016/j.preteyeres.2015.09.003.

[25] Rachitskaya AV, Yuan A. Argus II retinal prosthesis system: An update. *Ophthalmic Genetics* 2016 37(3):260–266. 10.3109/13816810.2015.1130152.

[26] Lemaire W, Benhouria M, Koua K, Besrour M, Gauthier LP, *et al.* Retinal Stimulator ASIC

Architecture Based on a Joint Power and Data Optical Link. *IEEE J. Solid-State Circuits* 2021, 56(7):2158–2170.

[27] Jepson LH, Hottowy P, Mathieson K, Gunning DE, Dabrowski W, *et al.* Focal electrical stimulation of major ganglion cell types in the primate retina for the design of visual prostheses. *Journal of Neuroscience* 2013 33(17):7194–7205. 10.1523/JNEUROSCI.4967-12.2013.

[28] Besrour M, Zitoun S, Lavoie J, Omrani T, Koua K, *et al.* Analog Spiking Neuron in 28 nm CMOS. In *2022 20th IEEE Interregional NEWCAS Conference (NEWCAS)*. 2022 pp. 148–152. 10.1109/NEWCAS52662.2022.9842088.

[29] Besrour M, Lavoie J, Omrani T, Martin-Hardy G, Koleibi ER, *et al.* Analog Spiking Neuron in CMOS 28 nm Towards Large-Scale Neuromorphic Processors, IEEE Transactions on Circuits and Systems for Artificial Intelligence (IEEE TCASAI) - Submitted January 2025 - (Under Review) - arxiv.org/abs/2408.07734.