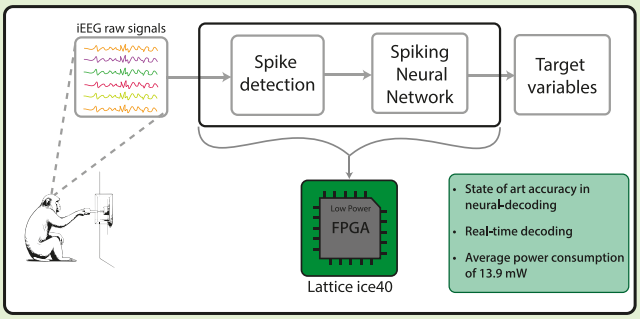
Low-Power FPGA-Based Spiking Neural Networks for Real-Time Decoding of Intracortical Neural Activity

Luca Martis, Gianluca Leone, Luigi Raffo, *Member, IEEE*, and Paolo Meloni, *Member, IEEE*

***Abstract*—Brain–machine interfaces (BMIs) are systems designed to decode neural signals and translate them into commands for external devices. Intracortical microelectrode arrays (MEAs) represent a significant advancement in this field, offering unprecedented spatial and temporal resolutions for monitoring brain activity. However, processing data from MEAs presents challenges due to high data rates and computing power requirements. To address these challenges, we propose a novel solution leveraging spiking neural networks (SNNs) that, due to their similarity to biological neural networks and their event-based nature, promise high compatibility with neural signals and low energy consumption. In this study, we introduce a real-time neural decoding system based on an SNN, deployed on a Lattice iCE40UP5k FPGA. This system is capable of reconstructing multiple target variables, related to the kinematics and kinetics of hand motion, from iEEG signals recorded by a 96-channel MEA. We evaluated the system using two different public datasets, achieving results similar to state-of-the-art neural decoders that use more complex deep learning models. This was obtained while maintaining an average power consumption of 13.9 mW and an average energy consumption per inference of 13.9 uJ.**



***Index Terms*— FPGA, low power, neural decoding, real time, spike detection, spiking neural network (SNN).**

I. Introduction

BRAIN–MACHINE interfaces enable direct communication between the brain and external devices. These interfaces are primarily dependent on a process commonly called *neural decoding*, which involves the processing of complex brain activity patterns, to convert thoughts or intentions into actionable outputs, such as controlling a computer cursor [1] or a prosthetic limb [2]. By bridging the gap between the human brain and technology, BMIs hold immense potential to restore lost functions [3], [4], [5], improve human capabilities, and deepen our understanding of the brain.

Received 2 October 2024; revised 14 October 2024; accepted 15 October 2024. Date of publication 1 November 2024; date of current version 13 December 2024. This work was supported in part by the European Union's Horizion 2020 Research and Innovation Program under Grant Agreement GA 101140052 (H2TRAIN) and in part by NextGenerationEU Mission 4, Component 2, Investment 1.5, CUP B83C22002820006—Project METBIOTEL—Innovation Ecosystem ECS 0000024 ROME TECHNOPOLE SPOKE 1, and SPOKE 6. This is an expanded paper from the 2023 IEEE Biomedical Circuits and Systems Conference (BioCAS), 2023, pp. 1–5 [DOI: 10.1109/BioCAS58349.2023.10389037]. The associate editor coordinating the review of this article and approving it for publication was Dr. Zhenghua Chen. *(Corresponding author: Luca Martis.)*

The authors are with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy (e-mail: luca.martis@unica.it; gianluca.leone94@unica.it; raffo@unica.it; paolo.meloni@unica.it).

Digital Object Identifier 10.1109/JSEN.2024.3487021

An important evolutionary path in the landscape of BMIs is provided by intracortical microelectrode arrays (MEAs), a fast-evolving technology that enables the monitoring of the brain activity with unprecedented spatial and temporal resolutions [6], [7] to improve the accuracy of the decoding. Several artificial intelligence (AI) algorithms have been shown to be effective in analyzing the low-frequency components of neural signals, commonly known as local field potential (LFP), or electrical *spikes* implementing the interaction between neurons, recorded in traces and referred to as multiunit activity (MUA). However, these algorithms are hard to deploy on embedded resource-constrained processing platforms, as real-time execution of such complex processing steps must be supported at very high rates, imposed by the typical sampling frequency and electrode count in MEAs. Thus, fully embedded neural decoding still needs to be further investigated, with the aim of integrating brain signal processing with data acquisition and on-site activation, improving responsiveness, reliability, power/energy, and communication efficiency.

A promising solution to this technology gap is offered by spiking neural networks (SNNs), a computational model that draws inspiration from biological neural networks, providing reduced power consumption and high compatibility with neural signals. SNNs are event-based algorithms: they operate on binary spikes and, thus, limit computing effort to be spent when actual activity happens, increasing power/energy efficiency. Furthermore, event-based information encoding is

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

intrinsically compatible with the processing of MUA acquired by MEAs.

The main goal of our research is to create an energy-efficient, flexible, and affordable BMI decoding system capable of extracting motor information from neural activity. With this aim, we combine SNNs with low-power FPGAs to achieve very low power consumption and, at the same time, adaptability to different experiments.

We have developed an integrated processing system, inspired by a base implementation proposed in [8], deployed on a Lattice iCE40UP5k FPGA, capable of extracting MUA from a 96-channel raw neural signal and decoding it in real time to obtain multiple motion-related variables with an adequately trained lightweight SNN. Our platform is the first, to the best of our knowledge, end-to-end near-sensor neural decoding system exploiting the potential of SNNs.

The main findings of this study can be summarized as follows.

1) We demonstrate the usability of lightweight SNNs for neural decoding, proposing an SNN that receives MEA-acquired MUA as input and decodes multiple target variables, related to the kinematics and kinetics of hand motion, in real time.
2) We propose an integrated processing system that includes, on the same low-power FPGA, the hardware extracting MUA from intracranial EEG signals, an SNN processor executing the SNN inference, and the circuitry required for input/output and system management.
3) We evaluate the accuracy of the system on two different datasets related to different decoding tasks, achieving results comparable to state-of-the-art methods.
4) We evaluate the computational efficiency of our solution, highlighting the potential of sparsity-aware on-FPGA SNN processing on this task, measuring an average power consumption of 13.9 mW.

The remainder of the article is organized as follows. Section II provides a brief overview of related works. Section III describes the two datasets used, the spike detection pipeline, the architecture of the SNN employed, and the training process along with the metrics used to evaluate the network's decoding accuracy. Section IV briefly outlines the hardware architecture used. Section V presents the results in terms of decoding accuracy and hardware performance, while Section VI compares these results with those of related works. Finally, Sections VII and VIII are dedicated to future work and conclusion, respectively.

II. Related Works

Using AI to decode neural signals is a very active topic in the literature. Neural networks and deep learning algorithms have been demonstrated to be very promising.

The studies [9], [10], [11], [12] have investigated different deep learning-based algorithms demonstrating their reliability in decoding motion intention using input features extracted from neural signals, such as MUA, single-unit activity (SUA), or entire spike activity (ESA).

The works [13] and [14] employ two new types of algorithms for decoding motor intentions from MUA. The work [13] achieves state-of-the-art accuracy, while the work [14] that utilizes the dataset [15] and three from a human with tetraplegia achieves state-of-the-art accuracy on the dataset [15] but lower accuracy when decoding neural signals from humans.

Studies [16], [17] utilized LFP signals to decode kinetics information. Specifically, [16] employed regularized linear discriminant analysis (RLDA) to classify four classes of grips and linear ridge regression (LRD) to decode the intensity of force. Meanwhile, [17] utilized partial least squares (PLS) for decoding the intensity of force.

In all mentioned works, research efforts focus on training and testing a decoding algorithm in offline settings, rather than guaranteeing real-time functionality in its implementation.

Our work tries to close this gap and focuses on SNNs as a prospective solution. Several works have used SNNs to reduce resource and power utilization by exploiting the event-driven nature of the task to allocate resources only when needed.

In [18] and [19], an SNN was used for classification tasks: one for image classification and the other for surface roughness classification. In both cases, the SNN achieved state-of-the-art accuracy and demonstrated suitability for real-time inference by leveraging the event-driven nature of the input.

These works only study the algorithm without analyzing its deployment on hardware.

In [20], [21], [22], [23], and [24], SNN-based approaches implemented on FPGA-based platforms are proposed, focusing on learning capabilities and accuracy optimization. The work in [20] deals with visual perception, in a task aimed at decision-making, with a focus on online learning. The approach presented in [21] also presents an online learning mechanism, tested on the MNIST dataset. In [22], the authors use SNNs to improve the performance of obstacle detection in connected vehicles. The approach in [23] tries to emulate the learning mechanism of hippocampus to show flexibility and robustness. In work [24], they implement a large-scale biologically meaningful neural network with one million neurons, capable of investigating the mechanisms of soma-to-dendritic interactions that are essential in neural systems. None of these works is, thus, specifically oriented to neural decoding. All these works rely on SNN accelerators based on bigger FPGAs, such as BiCoSS [25], deployed on a system composed of 35 Cyclone IV FPGA system, or LaCSNN [26], targeting Altera Stratix III, and, thus, do not focus on portability and wearability.

In [27], [28], and [29], SNNs were also applied to the neural decoding task. In [27], an SNN consisting of four dense layers was used to decode the speed of a fingertip from the spiking band power (i.e., the averaged intracortical signal in the 300–1000-Hz frequency band). In [28], SNNs have been shown to align with artificial neural networks and long short-term memory (LSTM) in accuracy while ensuring lower computational complexity, in terms of operations performed, memory accesses, and model size. Dethier et al. [29] have mapped a Kalman Filter on an SNN for neural decoding, achieving good task-related accuracy. However, in these studies, a practical implementation of the algorithms on an embedded processing system was not addressed.

Although numerous hardware solutions have been suggested for real-time intracortical neural signal processing, most of them only address spike sorting [30] or spike detection [31], [32], [33].

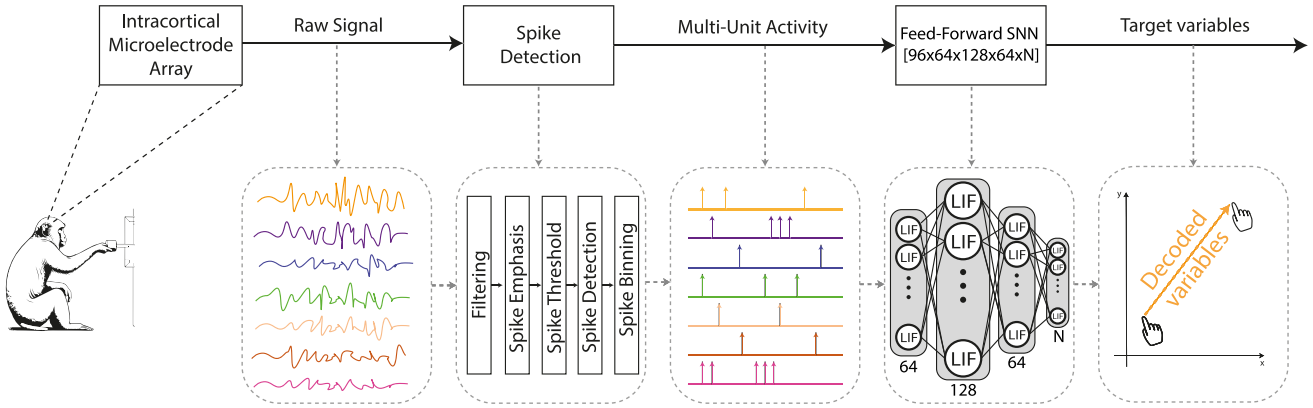


Fig. 1. General overview of the proposed approach. In the first step, we extract neural signals using an intracortical MEA. Next, we process the raw neural signals to obtain MUA, which serves as the input for the SNN. The SNN consists of four dense layers of leaky LIF neurons, where the membrane potentials of the network's last neurons are used to track the target variables related to the kinematics or kinetics of the performed movement.

To the best of our knowledge, only a few works offer a complete neural decoding system on an embedded device. The study presented in [34] introduces an ASIC that extracts spiking band power from the raw neural signal and applies a steady-state Kalman filter, enabling real-time prediction of finger movements using data from 93 channels acquired from a Utah MEA.

The works in [8], [35], [36], and [37] are the closest to ours.

A foundational work in this field is presented in [35], where a dedicated ASIC circuit implements an SNN-based decoder for a closed-loop experiment. However, despite the advanced application, the decoder does not encompass the entire processing chain, as feature extraction is offloaded to an FPGA.

Liao et al. [36] use microcontroller units (MCUs) as the target platform. Leveraging a mature and widely accessible technology, this approach achieves favorable power efficiency and offers broader adoption within the community compared to custom ASICs. However, it comes at the cost of slower execution times due to the limitations of general-purpose processors for SNN inference.

In [37], a two-layer dense SNN was used to decode the velocity displacement of a handle, moved by a monkey during a delayed reach-to-grasp task. A single variable is monitored, using a Zynq-7010 APSoC for implementation without more specific detail on power figures and related optimization.

Our work extends [8], where a four-layer dense SNN was utilized to decode not only the velocity displacement of the handle but also the finger pressure applied by the monkey. The system presented in [8] receives as input the raw neural signal derived from a 96-channel MEA and performs online spike detection, which will be the input for the SNN. The study described in [8] utilizes an Artix-7 FPGA that consumes 56.4 mW of power, leaving space for power efficiency enhancements.

In our present study, we take the approach one step further. We changed the platform from the Artix-7 FPGA to the much smaller and resource-constrained Lattice iCE40UP5k FPGA to achieve lower power consumption. To achieve this, we redesigned our SNN accelerator to match the capabilities of this low-power FPGA. We replaced the more complex MicroBlaze softcore with a simpler SErial RISC-V, and finally, we switched from 16- to 8-bit weight quantization. By optimizing the design to fit within the iCE40, which dissipates only a few milliwatts, comparable to a generic microcontroller, we moved to a drastically reduced power envelope. This shift opens entirely new possibilities for exploiting the system in highly wearable configurations.

Moreover, we tested the effectiveness of the network on an additional dataset related to a different motor task. Within the assessment, we also explored different training strategies, refining the achievable accuracy with minimal performance/resource overhead.

III. Methods

The objective of this study is to decode motor information from neural activity. Our overall approach to neural decoding is shown in Fig. 1.

The process followed a series of key steps. First, we selected two public datasets containing raw neural signals acquired during motor tasks. From these raw neural signals, we extracted MUA through a process known as spike detection. The extracted MUA will serve as the input for an SNN, which is trained to decode variables related to the kinematics or kinetics of the hand. The SNN was trained offline and evaluated using two common decoding accuracy metrics widely employed in the literature.

This section is organized according to the flow outlined previously: we begin by describing the two selected datasets (see Section III-A), followed by an explanation of the spike detection method used to extract MUA (see Section III-B). Next, we present the architecture of the SNN employed in our study (see Section III-C), and finally, we detail the training process of the SNN (see Section III-D), along with the metrics used to evaluate decoding accuracy (see Section III-E).

A. Datasets

We used two public datasets as reference (dataset I: [38] and dataset II: [15]) containing neural signals recorded during motor tasks from Rhesus macaque monkeys, where, in both cases, neural data were acquired using the Utah array [39], a sensor consisting of 96 electrodes spaced 400 μm apart, covering an area of 4 $\times$ 4 mm, and designed for intracortical applications.

In Fig. 2, an overview is presented of the contents of the two datasets, including the input, target, and the processing

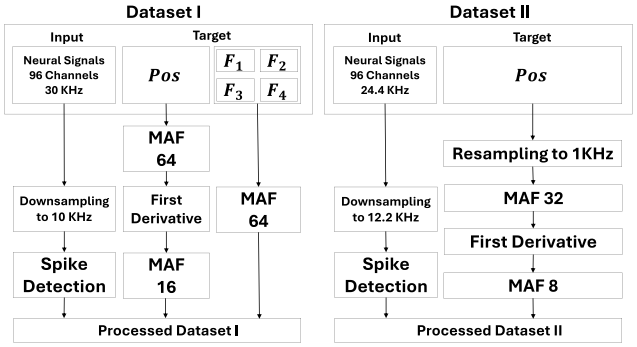


Fig. 2. Representation of the steps for preparing the datasets for training. On the left, the steps related to dataset I are illustrated, while, on the right, the steps for dataset II are illustrated. The term MAF N in the figure indicates filtering using an MAF of order *N*, while the first derivative is obtained by subtracting adjacent samples.

TABLE I
RECORDING OF DATASET II USED

Recording	Duration	Reaches
indy_20170124_01	9 min	485
indy_20170127_03	12 min	583
indy_20170131_02	13 min	635
indy_20160630_01	24 min	1023
indy_20160622_01	41 min	970

steps required to prepare the datasets for training. The steps are similar and mainly differ in the order of filters used to make the target variables smoother.

1) Dataset I: The dataset contains two recordings acquired from two different monkeys, identified as L and N. For this study, only the recording from Monkey N was used, which lasted approximately 16 min and included 160 trials. Neural signals were collected at a sampling rate of 30 kHz using the previously mentioned sensor. The recordings were obtained while the monkey performed an instructed delayed reach-to-grasp task, which involved pulling a cuboid handle.

In addition to the neural signals, the dataset includes the handle's position and the four applied forces, sampled at 1 kHz. It also provides MUA and SUA signals extracted using the methods described in [38]. In this study, neural signals were subsampled to a frequency of 10 kHz, and before the training phase, the handle position and the forces were smoothed using a moving average filter (MAF) of order 64. Finally, to calculate the handle velocity, we subtract adjacent samples of the smoothed position and then apply a moving average filter with an order of 16.

2) Dataset II: Data were acquired from two monkeys indicated as I ("indy") and L ("loco"). The dataset comprises 37 sessions from Monkey I (spanning approximately ten months) and ten sessions from Monkey L (spanning around one month). However, for this study, we analyzed only five recordings, as indicated in Table I.

Neural signals were collected at a sampling rate of 24.4 kHz using the same sensor as in Dataset I, positioned in the primary motor cortex, while the monkey reached for targets displayed as circles with a 5-mm radius on an 8×8 square grid. In addition to the neural signals, the dataset includes finger positions on the x- and y-axes and target locations, sampled at 250 Hz. It also provides MUA and SUA signals extracted using the methods described in [13]. In our study, neural signals were subsampled at 12.2 kHz, and behavioral data were resampled at 1 kHz. Before the training phase, the positions were smoothed with an MAF of order 32, and their first derivative was calculated by subtracting adjacent samples to derive velocity. Subsequently, the velocities were also smoothed using an MAF of order 8.

B. Spike Detection

The first step of the system is to convert the raw neural signal into MUA to be used as the input for spike detection. Each part of the algorithm was chosen to be as efficient as possible in terms of resource utilization.

The spike detection pipeline can be summarized as follows.

1) *Filter:* A second-order moving average difference (MAD) filter, described as follows, is applied to the raw signal to remove the low-frequency components:
$$\mathrm{MAD}(n) = x(n) - \frac{1}{2}\left[x(n-1) + x(n-2)\right]. \quad (1)$$
We selected this filter for two primary reasons: first, it can be implemented in a multiplierless way; second, it is memory-efficient since only two samples need to be stored.
2) *Spike Emphasis:* The absolute value of the filtered signals is used to emphasize the shape of the spike and enhance the detection reliability.
3) *Threshold:* The mean value of the rectified signal multiplied by 4 in the case of Dataset I and by 5 in the case of Dataset II, is used as the threshold above which a spike event is detected.
4) *Threshold Update:* The spike threshold is updated every 8912 samples.
5) *Spike Detection:* When the absolute value of the filtered signal exceeds the threshold, a spike is produced.
6) *Refractory Period:* Following the detection of a spike, a refractory period of 1 ms is implemented to prevent multiple detections of the same spike.
7) *Spike Bins:* Spikes are counted over a specific time interval of 1 ms for each channel.

In this configuration, the output frequency of the MUA is set to 1 kHz. Moreover, when the refractory period is the same as the bin window, the bins are limited to having values that are strictly either 1 or 0.

C. SNN Architecture

To define the SNN topology to be used in this work, we have performed a preliminary design exploration, comparing several alternatives in terms of achievable accuracy and computational complexity. As a main constraint, we have considered the memory availability on the target development platform. The final selected network topology, as shown in Table II, is the one reaching higher accuracy. Other more complex alternatives did not provide additional improvements. The chosen network is a feedforward SNN composed of four fully connected layers of leaky integrate-and-fire (LIF) neurons, which are modeled as follows:
$$U[t] = \beta U[t-1] + \sum ws[t] - S_{\mathrm{out}}[t-1]\theta$$
$$S_{\mathrm{out}}[t] = \begin{cases} 1, & \text{if } U[t] > \theta \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

TABLE II
NETWORK TOPOLOGY

Layer	Inputs	Neurons
1	96	64
2	64	128
3	128	64
4	64	$N_{targets}$

In this model, U represents the neuron's membrane potential, β is the membrane potential decay rate, w denotes the synaptic weights, s indicates the input spikes, S_{out} is the output spike, and θ represents the neuron's threshold. When the potential exceeds the threshold value θ, a spike S_{out} is generated and a reset mechanism is activated, subtracting the threshold value from the potential. As mentioned, the input to the network is MUA, while the membrane voltages of the neurons in the final layer of the network are used as outputs. This allows the SNN to predict continuous-valued target variables. For this purpose, the number of neurons in the final layer N_{targets} depends on the number of variables to be decoded. Specifically, using Dataset I, since we decode the speed and the four forces applied to the handle, the number of neurons is five. Conversely, using Dataset II, as we decode the speed along the x- and y-axes, the topology includes two output neurons.

Since the potential of the last neurons directly serves as the output of the network, the reset mechanism has been disabled in the final layer.

D. Training Scheme

We used the snnTorch[1] package [40] to train the network, which utilizes backpropagation through time (BPTT) as the training algorithm. The training strategy was also selected after a preliminary exploration aimed at identifying the benefits provided by different settings and hyperparameters.

Fig. 3 shows the different tests performed on a single recording of Dataset II, which were conducted before arriving at the final configuration, which was then used for all the recordings of both datasets.

Specifically, eight different tests were performed.

1) *w:* only the synaptic weights have been trained.
2) *w* and β*:* The synaptic weights and the decay rate of each neuron have been trained.
3) *w* and θ*:* The synaptic weights and the threshold of each neuron have been trained.
4) *w,* θ*,* and β*:* The synaptic weights, the decay rate, and the threshold of each neuron have been trained.

For each configuration, the voltage reset method was investigated. Two approaches were tested: setting the voltage to 0 and subtracting the threshold voltage from the prespike voltage. Generally, it can be observed that the primary contribution to accuracy improvement comes from making the decay rate trainable. Conversely, making the threshold trainable does not lead to significant improvements. It is also noticeable that using a subtraction-based reset mechanism consistently yields better results with the same trained parameters.

The final training strategy, therefore, involved making each neuron's membrane potential decay rate β trainable and utilizing a reset method that subtracts the threshold value from the potential, instead of resetting it to 0, while keeping the threshold fixed at a constant value.

[1]https://snntorch.readthedocs.io/en/latest/

TABLE III
TRAINING SETTINGS AND PARAMETER QUANTIZATION

Hyperparameter	Value	Parameter	Width
Epochs	100	Weights	8-bit
Batch size	10		
Learning rate	1e-3	Potential	32-bit
Loss	MSE		
θ	0.1	Decay	13-bit
β	Learned		

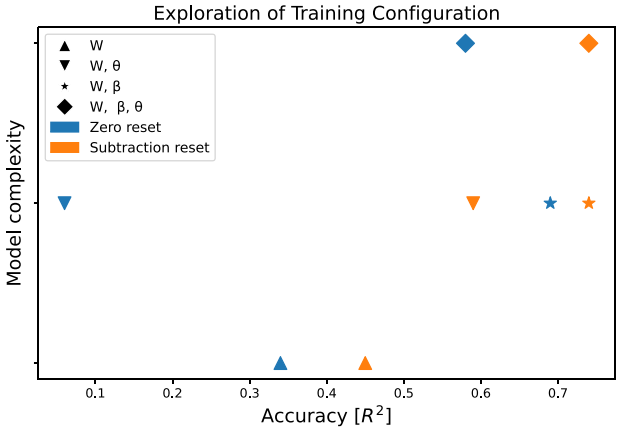


Fig. 3. Results of the exploration conducted to identify the best training configuration.

During training, the loss function was defined as the mean squared error (MSE) between the membrane potential of the neurons in the last layer and the target variables. Adam is chosen as the optimizer, and fast sigmoid is applied as a surrogate function.

Each dataset is partitioned based on the number of tasks performed by the subject, rather than the duration of the recording. The training set comprises 80% of the tasks completed, while the validation set and test sets each contain 10%. The number of epochs was set to 100, the batch size to 10, and the learning rate to $1e^{-3}$.

After training, we selected the network that exhibited the lowest loss in the validation set as our best model. Training for each recording was performed by segmenting the training set into windows using the indications provided in the dataset, marking the beginning and end of a task. Thus, the starting point of different windows corresponds to a zero crossing in the target variable waveform. Finally, the network was tested on continuous recordings without segmentation. It is possible to access the algorithm used for training via the following GitHub link https://github.com/LucaMartis00/SNN-Training-scheme.git.

Once the model was trained, it was quantized to meet the hardware constraints, as explained in more detail in Section IV. Table III summarizes the relevant hyperparameters utilized and the formats used to represent the network parameters. Specifically, the membrane potential is represented in a 32-bit fixed-point format to prevent saturation, while the decay rates are stored in a 13-bit fixed-point format and the weights in an 8-bit fixed-point format.

E. Metrics

To evaluate the performance of the network, we use two different metrics: Pearson's correlation coefficient (CC) described

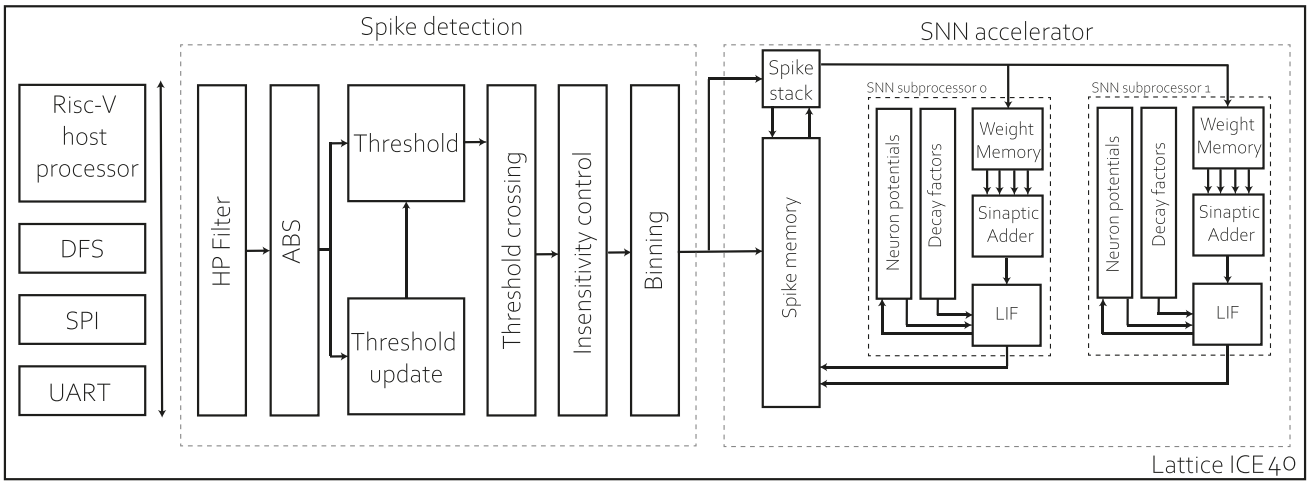


Fig. 4. System architecture is implemented on a Lattice iCE40UP5k FPGA and features an RISC-V softcore for system management and control. Key components include a DFS module for selective hardware module gating, along with SPI and UART interfaces. The spike detection module converts raw neural signals into MUA. The raw neural signal undergoes filtering and rectification, followed by threshold crossing for spike identification. After detection, a 1-ms refractory period prevents further detections on the same channel. The binning module aggregates detected spikes per millisecond on each channel, providing input to the SNN accelerator. The SNN accelerator performs inferences using a two-way parallel SNN subprocessor. This subprocessor consists of a weight memory, a four-way synaptic adder for accumulating active synapse weights, and a neuron module for updating the LIF neuron state. In addition, the SNN accelerator includes a spike memory to buffer partial results from layer executions and a spike stack to keep track of the active spikes.

in (3) and the coefficient of determination (R^2) described by (4)

$$CC = \frac{\sum_{i=1}^{N} (y_i - \bar{y}) \left(\hat{y}_i - \bar{\hat{y}}\right)}{\sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2} \sqrt{\sum_{i=1}^{N} \left(\hat{y}_i - \bar{\hat{y}}\right)^2}} \tag{3}$$

$$[2.5\ mm]\ R^2 = 1 - \frac{\sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2} \tag{4}$$

where N is the number of samples, y_i are the target values, $\bar{y}$ is the mean of target values, $\hat{y}_i$ are the predicted values, and $\bar{\hat{y}}$ is the mean of predicted values. The metrics were computed separately for each output variable: on velocity and four forces for dataset I and on velocity on x any axis for dataset II.

IV. System Architecture

In order to prove the usability of an SNN-based approach to neural decoding in real time and in a very low-power envelope, coherent with wearable or prospectively implantable setups, we have deployed the previously described decoding strategy on an integrated processing system. The system architecture, as shown in Fig. 4, is implemented on a Lattice iCE40UP5k FPGA and comprises of the following:

1) an RISC-V softcore responsible for system management and control;
2) a dynamic frequency scaling (DFS) module, enabling selective gating of hardware modules;
3) SPI and UART interfaces;
4) spike detection and binning modules processing raw iEEG signals to generate MUA-based bins;
5) an SNN accelerator receiving MUA bins as input and decoding N_{targets} variables (five in the case of Dataset I and two in the case of Dataset II).

To evaluate real-time functionality and measure system power consumption, we utilized a flash memory connected to the SPI interface. This configuration emulates the acquisition of sample data from a sensor array. Finally, the decoding results were transmitted to a PC through the UART interface.

This section is divided into three subsections, each describing the main components of the system. In Section IV-A, the architecture of the spike detection and binning module is presented. In Section IV-B, the architecture of the module implementing the SNN is detailed. Finally, Section IV-C describes the processor used to manage the system.

A. Spike Detection and Binning Modules

Spike detection and binning are performed sequentially for MEA channels: the former extracts intracortical spikes from raw neural activity, and the latter sums the spikes in each millisecond (binned activity).

Spike detection comprises raw neural signal filtering, rectification, and comparison with a dynamic threshold to assess the presence of a spike. After every spike detection, the channel is set in a refractory period, inhibiting further detections for 1 ms. Finally, the spike binning module sums the spikes per millisecond per channel generating the SNN decoder inputs.

Aiming for resource utilization reduction, the filter is designed to be multiplierless, as well as all the other modules of the spike detection and binning; therefore, the low-frequency components are removed by using a MAD filter, which subtracts from the input sample the output of a second-order MAF, as in (1). The implementation only requires a BRAM to store two successive samples per channel (Lattice iCE40UP5K integrates 4-kb BRAM macros) and a few LUTs for addition, subtraction, and right shift in (1).

This module is also in charge of computing threshold adaptation, which is obtained by averaging the rectified signal over windows of 8192 samples and multiplying them for an integer constant using sums and shifts.

The channel insensitivity period that follows every spike detection is implemented through BRAM-based counters that keep track of time steps after a spike to deactivate the detection for 1 ms.

Since the refractory period is equal to the binning window, the value of the bin cannot be higher than 1. Therefore, the binning module comprehends a 96-bit register, where every bit is associated with a channel, and a 4-bit counter to keep track of the time (ten time steps). Bins are obtained with a logical OR between the incoming spike and the bit associated with the same channel in the 96-bit register. At the end of the 1 ms window, the bins register is streamed out to the SNN accelerator and reset.

B. SNN Hardware Accelerator

The SNN accelerator is also designed to optimize resource and energy efficiency. It executes inference by processing the binned neural activity through a two-way parallel SNN subprocessor. Each SNN subprocessor is reused over time, in different clock cycles, to process all the LIF neurons in the SNN computing (2). The SNN subprocessor comprises a weight memory, accessed four words at a time, a four-way synaptic adder used to accumulate the active synapse weights, and two BRAM-based FIFOs that store, respectively, the neuron potentials and the decay factors; finally, a neuron module is used to update the LIF neuron state. Moreover, the SNN accelerator embeds a spike memory and a spike stack, which serve, respectively, as a buffer for storing partial results derived from the layers' execution and exploiting the sparsity of the inputs.

1) SNN Subprocessor: The weight memories are mapped in bigger memory macros, i.e., single-port RAMs (SPRAMs). The target FPGA embeds four SPRAMs of 256 kb, with a 16-bit read/write port each. The size of the trained models does not require using all the storage in the four macros; however, using them in parallel permits to sustain a read throughput of up to eight weights per clock cycle, four per weight memory.

To achieve this reading throughput, it was necessary to switch to an 8-bit fixed-point representation of the weights. Thus, the SNN model was quantized to enable efficient deployment on the designated hardware system and optimize memory usage. We have reduced the weight resolution to 8 bits by applying adequate observers during training to identify the dynamics and the scaling factors to be used. Quantization did not determine any significant accuracy loss.

Most of the processing load is carried out by the four-way SIMDs synaptic adders, implemented using LCs, which compute the synaptic current in input to the LIF neurons by accumulating four synaptic weights per clock cycle each. The two adders operate individually on different neurons of the same layer. The currents are used to integrate the LIF neurons in the LIF modules. Each LIF module stores the neuron potentials and the decay factor in BRAM-based FIFO, cyclically multiplies the potential value by the decay factor employing a DSP-mapped 16 $\times$ 16 multiplier, and sums it to the synaptic current. The membrane potential is represented in a 32-bit fixed-point format to prevent saturation, caused by decay rate values nearing 1, while decay rates are stored with a 13-bit fixed-point representation.

Then, an LC-based comparator is used to verify if the potential value exceeds the threshold. If this is true, the threshold value is subtracted from the neuron potential, and a spike is generated, following the rules stated by (2).

2) Spike Memory: The spike memory is implemented as a dual BRAM-based buffer. Since it is read once per neuron of the layer, its content must be stable for the whole inference execution. Therefore, a double ping–pong buggering strategy is used. To create a more compact implementation, two 4-kb BRAMs have been used for this purpose.

3) Spike Stack: One of the primary features of an SNN processing engine, as seen in most neuromorphic processors, is the ability to take advantage of event sparsity by performing necessary computations solely when spike events initiate them. To implement this, we have added a stack unit, in charge of monitoring spiking activity, at every time step, neuron by neuron. During the integration of the LIF neurons, the addresses of firing neurons are annotated in the stack. Thus, during current computation, the stack keeps track of which active weights need to be retrieved from the weight memory, along with their respective addresses. Weights that are not activated by previous events (i.e., corresponding to nonfiring neurons) are excluded from the stack, thereby skipping the corresponding computations and leveraging sparsity. The stack is also implemented as a dual buffer to enable events to be produced and consumed independently in different inference steps. The saved processing cycles can be used to improve energy efficiency by scaling the frequency to set the system in idle mode.

C. RISC-V Softcore

In order to increase flexibility and ease of use, our platform integrates an RISC-V processor softcore named SErial RISC-V (SERV)[2] into the architecture. SERV is designed to be small, efficient, and simple; thus, its use enables us to keep resource requirements compatible with low-power FPGAs. Considering that the core delegates the SNN accelerator for more performance-hungry tasks, this parsimonious implementation does not compromise the performance. SERV provides the following capabilities.

1) Loading synaptic weights from the SPI flash drive and storing them in the on-chip memory, this is very useful to adapt to different SNN topologies without changing the hardware configuration.
2) Loading the input samples from the SPI flash drive in real-time enabling the acquisition system emulation (using a programmable core for this purpose enables the change of input data rates and formats via firmware).
3) Finely managing power consumption: the processor can be used to set the system in low-power mode via clock gating and frequency scaling, when required by the application.

V. Experimental Results

In this section, we present the results that we have achieved. We have divided the section into three subsections. Section V-A presents the accuracy achieved by the SNN in neural decoding. We also analyze the effectiveness of our spike detection method by comparing our results with the accuracy obtained using the spike activity labeled in the dataset. Section V-B discusses the model's ability to exploit spike sparsity, from both a software and hardware perspective.

[2]https://github.com/olofk/serv

TABLE IV
SNN DECODING ACCURACY

Dataset	Recording	Target	GT		Spike Detection	
			CC	R^2	CC	R^2
I	Monkey N	V	0.82	0.66	0.87	0.71
		F	0.87	0.76	0.87	0.76
II	indy_20170124_01	V	0.87	0.74	0.87	0.74
II	indy_20170127_03	V	0.86	0.72	0.86	0.74
II	indy_20170131_02	V	0.85	0.72	0.86	0.72
II	indy_20160630_01	V	0.76	0.57	0.67	0.44
II	indy_20160622_01	V	0.86	0.72	0.85	0.71

Finally, we present the hardware results in terms of FPGA resource usage, throughput, and power dissipation.

A. Decoding Accuracy

The network was trained and tested using the result of our spike detection. In addition, since other works focusing solely on decoding utilize the spike detection included in the datasets (GT), we trained and tested the network also using the provided spike detection to ensure a fair comparison with such works and to compare our spike detection, suitable for our hardware, with a more sophisticated one.

Table IV presents the accuracy results obtained. Specifically, the first part displays the accuracy results achieved on Dataset I, showing the accuracy value obtained for speed and the average accuracy results across the four forces. The second part reports the accuracy results obtained for individual recordings from Dataset II. It includes the average speed calculated separately for the x- and y-axes.

The accuracy results obtained from the SNN confirm the validity of our spike detection. In most cases, the accuracy achieved using our spike detection is comparable to or even superior to the spike detection provided in the dataset, obtained with more complex methods. The only case where it is outperformed is the registration indy_20160630_01. However, in this case, the registration seems to be particularly noisy and difficult to decode. There is a noticeable decline in performance compared to other registrations, a finding that is also supported by other research [10], [11], [41].

Finally, in Fig. 5, the output of the decoder is compared to the ground truth, illustrating that the decoder is capable of reliably tracking target variables. A detailed comparison with state-of-the-art overall accuracy is demanded in Section VI.

B. Sparsity Exploitation

In this section, we evaluate the capability of our system to exploit the sparsity of spikes through the spike stack. Our approach takes into account the event-driven nature of the algorithm, performing additions and weight-memory accesses only if needed. However, since, with the aim of limiting the overhead introduced by the stack, inputs are grouped in sets of four channels, partially active groups can occur, resulting in some unnecessary operations and reducing the number of operations saved compared to the ideal case, as shown in Fig. 6. To quantify the effectiveness of this approach, the number of operations saved was measured for each of the trained models during the inference of the test set and compared to the ideal case where all unnecessary operations are skipped. On average, it was found that the mean number

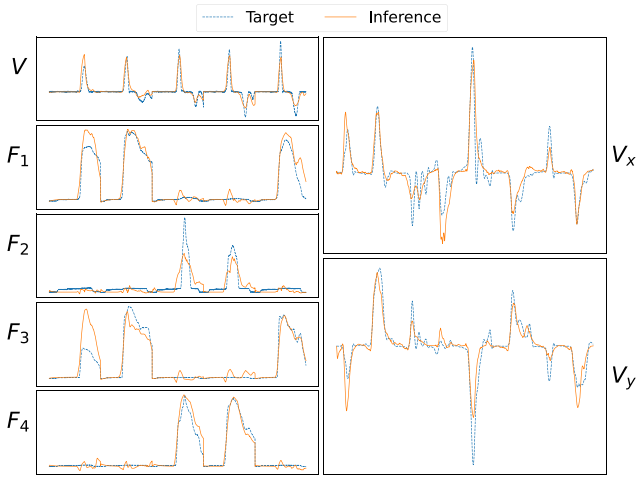


Fig. 5. Comparison between the output of the SNN (inference) and the target variables (target). On the left, the graph shows the five outputs obtained from a portion of the test set of dataset I. Specifically, from top to bottom, it illustrates the speed and the four forces applied to the handlebar by the monkey. On the right, the graph shows the two outputs obtained from a segment of the test set related to the recording indy_20170124_01 of dataset II. Specifically, from top to bottom, it illustrates the speed on the x-axis and the speed on the y-axis.

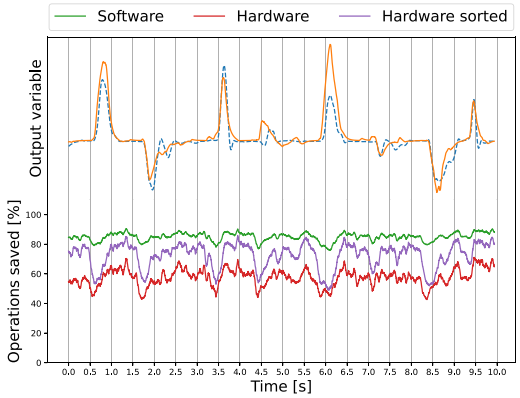


Fig. 6. Number of unnecessary skipped operations computed during test set inference.

TABLE V
RESOURCE USAGE ON LATTICE ICE40UP5K

LC	DSP	BRAM	SPRAM
4484 (84%)	4 (50%)	22 (73%)	4 (100%)

of unnecessary skipped operations in the ideal case is 88.25%, while, for our hardware, it is 65.84%. However, it is possible to sort the neurons that generate spikes more frequently to be adjacent, thereby reducing the number of unnecessary operations performed. Considering this case, the mean number of unnecessary skipped operations increases to 73.3%.

C. Hardware Performance

1) Resource Utilization: The Lattice iCE40UP5k FPGA contains 5280 logic cells (LCs), each of which includes one four-input lookup table (LUT4) and one flip-flop, 30 4-kbit embedded block RAMs (BRAMs), four 256-kbit SPRAMs, and eight DSPs.

Table V shows the hardware resources required by the current configuration; 84% LCs are used, and all available SPRAMs are used to store weights. However, it should be noted that they are not full, as the total occupied memory

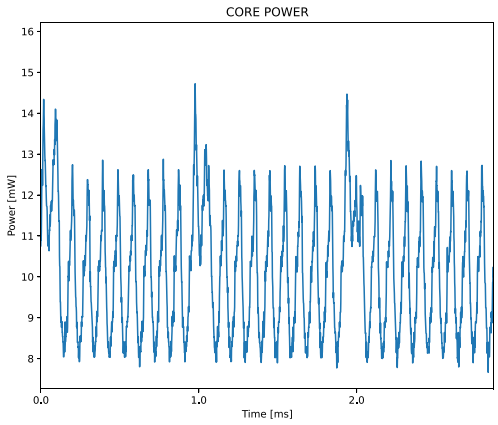


Fig. 7. Power dissipated during inferences by the FPGA core.

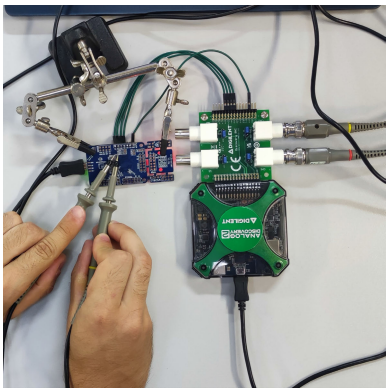


Fig. 8. Measurement setup used. The image shows the measurement of the voltage drop across the resistor connected to the core power supply.

is less than 20%. Four DSPs are used to accelerate neuron integration during inference, and 22 BRAMs are used to store the spikes generated, the membrane potential, the membrane potential decay rates, and the softcore firmware. These utilization figures highlight that our architecture exploits quite effectively the target platform, using most of the available resources and, thus, taking profit from most of its computational capabilities.

2) Inference Time: The implemented system supports up to 128 input channels, with the decoder clock frequency set at 22 MHz to ensure decoding times of less than 1 ms, thereby guaranteeing the possibility of doing real-time decoding.

More specifically, input acquisition/preprocessing and output transmission can overlap with SNN execution. Inside the SNN accelerator, different modules can also be executed as a pipeline, where the throughput is limited by the synaptic current computation. Since the hardware can perform eight-synaptic additions per clock cycle, one inference can be executed every T_{inf}, expressed as

$$T_{\text{inf}} = \frac{N_{\text{synapse}}}{8} T_{\text{clk}} \tag{5}$$

where N_{synapse} is the number of active synapses and T_{clk} is the clock period. In the worst case scenario, where all the synapses are active, T_{inf} is equal to 0.13 ms. However, taking into account the sparsity of spikes, it decreases to 0,03 ms.

3) Power Consumption: In the Lattice iCE40UP5k FPGA, there are three power supplies: VCORE provides 1.2 V for the internal FPGA components, while VCCIO0&1 and VCCIO2 supply 3.3 V for the I/O pins. Power dissipation was measured by connecting a resistor with a nominal value of $3.3 \pm 0.033\ \Omega$ to each power supply and measuring the voltage drop across them. The overall dissipated power is found to be 13.9 mW, with 10.1 mW dissipated by the core and 3.8 mW dissipated by the I/O.

Fig. 7 illustrates the real-time power dissipation of the core, recorded using a Digilent Analog Discovery 2 oscilloscope. It is observable that the peak power repeats every millisecond during inference. In contrast, during periods of SNN accelerator inactivity, the power dissipation is lower. The smaller and more frequent peaks correspond to data transmission, spike detection, and binning, occurring much more frequently, every 0.1 ms.

In particular, due to the overall low computational load of the softcore and the embedded DFS module in the system, we were able to dynamically gate the softcore, resulting in a power savings of 1.9 mW.

Fig. 8 shows the measurement setup used, in particular the measurement of the voltage drop across the resistor connected to the core power supply. The reported power consumption indicates that combining low-power FPGA and SNNs for neural decoding can be an effective solution. This approach can meet the stringent power constraints associated with battery-operated and highly portable implementations of this task.

VI. Comparison With State of Art

In this section, we have compared our work with others in the literature.

We divided the comparison into two subsections: the first focuses solely on decoding accuracy, while the second evaluates hardware performance. Specifically, Table VI compares our work with related approaches for the same task found in the literature. The table is organized as follows: the first part reviews works that utilize dataset I, the second part examines works using dataset II, and the final part covers works that use datasets different from ours. The table provides details on the input signals, the decoding algorithms, the platforms used for implementation, the model size (expressed in the number of parameters), and the achieved accuracy. The accuracy is reported as the mean CC for velocity (V_{CC}), the mean CC for force (F_{CC}), the mean coefficient of determination for velocity (V_{R^2}), and the mean coefficient of determination for force (F_{R^2}).

Table VII further presents the main characteristics of the hardware accelerator, including its power consumption, energy per inference, and inference execution time.

A. Decoding Accuracy Comparison

Regarding the works that have used the dataset I, we can see that using our spike detection, we achieve better results compared to all other works, except in the case of [12], where it is pertinent to note that their model uses 19 times the number of parameters that we use. Furthermore, their system does not function in real time and decodes only velocity, whereas we decode both velocity and the four forces.

When considering the studies that used dataset II, compared to [9], [10], [11], our performance appears slightly inferior.

TABLE VI
COMPARISON OF DECODING ACCURACY WITH RELATED WORKS

Work	Dataset	Signal	Decoder	Platform	Parameters	V_{CC}	V_{R^2}	F_{CC}	F_{R^2}
This work[3]	Dataset I	MUA	SNN	iCE40UP5k FPGA	22.6k	0.87 (0.82)	0.71 (0.66)	0.87 (0.87)	0.76 (0.76)
[37] 2023	Dataset I	MUA	SNN	Zynq-7010 FPGA	56.0k	0.83	-	-	-
[11] 2021	Dataset I	MUA	QRNN	PC	-	0.84	-	-	-
[12] 2021	Dataset I	SUA	RNN	PC	420k	0.91	-	-	-
This work[3]	Dataset II	MUA	SNN	iCE40UP5k FPGA	22.6k	0.82 (0.84)	0.67 (0.69)	-	-
[28] 2023	Dataset II	MUA	SNN	PC	6.33k[4]	-	0.65	-	-
[41] 2023	Dataset II	MUA	SNN	PC	4.9k[4]	0.77	0.59	-	-
[11] 2021	Dataset II	MUA	QRNN	PC	-	0.84	-	-	-
[9] 2019	Dataset II	MUA	LSTM	PC	-	0.84[5]	-	-	-
[10] 2022	Dataset II	MUA	QRNN	PC	327k	0.85	-	-	-
[10] 2022	Dataset II	MUA	LSTM	PC	341k	0.84	-	-	-
[10] 2022	Dataset II	MUA	MLP	PC	155k	0.82	-	-	-
[13] 2018	Dataset II	MUA	rEFH	PC	-	-	0.63	-	-
[14] 2022	Dataset II	LPF	EvoEnsemble	PC	-	0.76	-	-	-
[12] 2021	Dataset II	SUA	GRU	PC	1190k	-	0.76	-	-
[33] 2023	Dataset II	MUA	RM-LSTM	PC	-	$<$0.8	-	-	-
[27] 2022	Other dataset	SBP	SNN	PC	157k[4]	0.75	-	-	-
[17] 2016	Other dataset	LPF	PLS	PC	-	-	-	0.68	0.45

[3] In parentheses are reported the accuracy results obtained with spike detection included in the dataset.
[4] Derived from the topology of the network described in the article. [5] Derived from the results presented in the article.

TABLE VII
COMPARISON OF HARDWARE IMPLEMENTATIONS

Work	Year	Channels	Sample rate [kHz]	Feature Extraction	Task	Platform	Process [nm]	Decoder	Power [mw]	Energy/inf. [uJ]	Time/inf. [ms]
This work	2024	96	10 / 12.2	MUA	Hand Kinetics and kinematics	FPGA	40	SNN	13.9	0.417	0.03
[34]	2022	93	2	SBP	Finger 2D	ASIC	180	SSKF	0.588	-	-
[36]	2024	96	2	N/A	Finger 2D	MCU	22	SNN	0.5	1.88	0.12
[35]	2016	15	25	N/A	Object control	ASIC	180	SNN	4	-	-

However, the complexity of our networks is significantly lower. Moreover, as previously noticed, on this dataset, our spike detection is less effective. If we apply our decoding mechanism to the same spike detection results provided in the dataset, we achieve similar or higher accuracy, except in [10], when the QRNN is used.

Zhou et al. [28] complement the discussion by reporting accuracy obtained by filtering the network output. However, to permit a direct comparison with our algorithm, only the accuracy result achieved using the network without the output filter has been included in the table. Nonetheless, even after implementing the output filter, R^2 of 0.68 is obtained, slightly surpassing the results from our spike detection method, but still falling short in comparison to our network trained and evaluated using the spike detection included in the dataset. In addition, the filter utilized cannot operate in real time since the whole waveform is needed before processing begins.

The work [41] provides six trained networks[3] with six different recordings from dataset II, specifically three related to the monkey Indy, which we also used, and three belonging to the monkey Loco. Since, for the monkey Loco, 192 are used in input channels, which are not supported by our hardware, we considered only the three recordings related to the monkey Indy in the accuracy calculation, and the average of the three obtained accuracy values was included in the table. As observed, we achieve better accuracy whether using the spike detection provided in the dataset or our spike detection method. Compared to the work in [28] and [41], we achieve better results using a larger network. However, it should be noted that the size of our network was not chosen to be the smallest possible, but rather the best network that fits the capabilities of our hardware.

[3]https://github.com/NeuroBench/neurobench/tree/main/neurobench/examples/primate_reaching/model_data

The work presented [12] reports a 10% higher accuracy value. However, it is important to note that the network architecture in such work is significantly larger, using 54 times more parameters than ours.

The study [33] presents a hardware implementation for real-time MUA extraction. However, the decoding algorithm, based on LSTM, was implemented offline. A synthetic accuracy result is not provided, but, from the graphs, it appears that the CC falls below 0.8. This is lower than the accuracy achieved by our system, which implements both spike detection and the decoding algorithm in hardware.

Finally, although a direct comparison with the last two studies is not possible, since they use a different dataset, we can see that we achieve higher accuracy for both velocity decoding and force decoding. Summarizing, comparison with state-of-the-art approaches shows that our solution provides a good tradeoff between power efficiency, implementability, and accuracy. In general, the comparison suggests that our accuracy could benefit from a similarly affordable but more precise spike detection, which will be the object of future work.

B. Hardware Performance Comparison

Compared to the work in [36], our system exhibits higher power consumption, primarily because it employs a microcontroller, which has been a well-established technology for years and, in this case, utilizes a 22-nm process. However, their system takes four times longer than ours to perform an inference, which ultimately results in higher energy consumption per inference compared to our solution. Compared to [34] and [35], our system exhibits higher power consumption. However, since other metrics are not provided, a comprehensive comparison is difficult. Moreover, the power consumption reported in [35] accounts only for the SNN, excluding the feature extraction, which is performed externally by an FPGA.

In summary, our approach leverages the capabilities of a lightweight SNN that is optimally suited for the low-power FPGA that we have chosen. This enables us to achieve reduced energy consumption compared to our competitors while maintaining high throughput and accuracy.

Preliminary exploration indicates that using larger networks does not significantly improve accuracy. However, larger networks may be essential for processing larger MEAs. In such cases, we can implement the system on bigger FPGAs, which could impact power efficiency. Nevertheless, a more extensive configuration of the SNN processor would enable greater parallelism, improving latency and potentially offsetting energy consumption.

VII. Future Works

While this study presents a promising foundation, several limitations necessitate further investigation. The current system's accuracy, while encouraging, can be enhanced by expanding the training dataset. However, the scarcity of publicly available datasets and the computational demands of larger models pose significant challenges. Furthermore, assessing the generalizability of our approach across different subjects is crucial for practical applications. Future studies should focus on evaluating the system's performance on intersubject use cases and assessing its robustness to noise. Integrating online learning into the system's architecture could allow for more dynamic adaptation to individual users and evolving conditions, thereby enhancing both its versatility and robustness. In terms of power consumption, still, the approach still pays for a significant amount of quiescent power, which derives, first, from FPGA technology and second from a considerable contribution from IO and spike detection. Switching to an application-specific integrated circuit (ASIC) implementation could potentially lead to significant reductions in power consumption. In addition, optimizing channel selection through offline calibration of the dataset could help reduce power consumption by focusing on the most informative channels and minimizing the processing required for less relevant ones.

VIII. Conclusion

In this study, we have introduced a resource- and power-efficient intracortical BMI designed for concurrent and continuous neural decoding of multiple target variables associated with the kinetics or kinematics of a hand. The system is deployed on a Lattice iCE40UP5k FPGA and hosts a multiplierless spike detection pipeline to extract intracortical spikes, an SNN model that directly processes neural spikes inferring the values of the target variables, and an RISC-V softcore responsible for system management and control. The system achieved accuracy comparable to other works in the literature for a delayed reach-to-grasp task, for a free-reaching task, and in decoding the finger forces applied to a handle. While providing satisfying accuracy, the complexity of our model is generally lower than that of state-of-the-art approaches. Our event-based approach, combined with a low-power technology target, brings the power consumption on the chip to only 13.9 mW, paving the way to its use in a wide scope of experiments requiring high portability and long battery lifetime.

References

[1] C. Pandarinath et al., "High performance communication by people with paralysis using an intracortical brain-computer interface," *eLife*, vol. 6, Feb. 2017, Art. no. e18554.

[2] F. M. Petrini et al., "Six-month assessment of a hand prosthesis with intraneural tactile feedback," *Ann. Neurol.*, vol. 85, no. 1, pp. 137–154, Dec. 2018.

[3] X. Liu et al., "A fully integrated sensor-brain-machine interface system for restoring somatosensation," *IEEE Sensors J.*, vol. 21, no. 4, pp. 4764–4775, Feb. 2021.

[4] A. B. Ajiboye et al., "Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: A proof-of-concept demonstration," *Lancet*, vol. 389, no. 10081, pp. 1821–1830, 2017.

[5] M. Śliwowski, M. Martin, A. Souloumiac, P. Blanchart, and T. Aksenova, "Decoding ECoG signal into 3D hand translation using deep learning," *J. Neural Eng.*, vol. 19, no. 2, Mar. 2022, Art. no. 026023, doi: 10.1088/1741-2552/ac5d69.

[6] F. Boi et al., "Multi-shanks SiNAPS active pixel sensor CMOS probe: 1024 simultaneously recording channels for high-density intracortical brain mapping," *BioRxiv*, Aug. 1024, Art. no. 749911, doi: 10.1101/749911.

[7] G. N. Angotzi et al., "SiNAPS: An implantable active pixel sensor CMOS-probe for simultaneous large-scale neural recordings," *Biosensors Bioelectron.*, vol. 126, pp. 355–364, Feb. 2019.

[8] G. Leone, L. Martis, L. Raffo, and P. Meloni, "Spiking neural networks for integrated reach-to-grasp decoding on FPGAs," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2023, pp. 1–5.

[9] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Decoding hand kinematics from local field potentials using long short-term memory (LSTM) network," in *Proc. 9th Int. IEEE/EMBS Conf. Neural Eng. (NER)*, Mar. 2019, pp. 415–419.

[10] N. Ahmadi, T. Adiono, A. Purwarianti, T. G. Constandinou, and C.-S. Bouganis, "Improved spike-based brain-machine interface using Bayesian adaptive kernel smoother and deep learning," *IEEE Access*, vol. 10, pp. 29341–29356, 2022.

[11] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning," *J. Neural Eng.*, vol. 18, no. 2, Apr. 2021, Art. no. 026011.

[12] S.-H. Yang, J.-W. Huang, C.-J. Huang, P.-H. Chiu, H.-Y. Lai, and Y.-Y. Chen, "Selection of essential neural activity timesteps for intracortical brain–computer interface based on recurrent neural network," *Sensors*, vol. 21, no. 19, p. 6372, Sep. 2021.

[13] J. G. Makin, J. E. O'Doherty, M. M. B. Cardoso, and P. N. Sabes, "Superior arm-movement decoding from cortex with a new, unsupervised-learning algorithm," *J. Neural Eng.*, vol. 15, no. 2, Jan. 2018, Art. no. 026010, doi: 10.1088/1741-2552/aa9e95.

[14] X. Zhu, Y. Qi, G. Pan, and Y. Wang, "Tracking functional changes in nonstationary signals with evolutionary ensemble Bayesian model for robust neural decoding," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Red Hook, NY, USA: Curran Associates, 2022, pp. 22576–22588. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/8dcc306a2522c60a78f047ab8739e631-Paper-Conference.pdf

[15] J. E. O'Doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes, "Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology," Zenodo, Sabes Lab, Univ. California, San Francisco, CA, USA, Tech. Rep., May 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3854034

[16] T. Milekovic, W. Truccolo, S. Grün, A. Riehle, and T. Brochier, "Local field potentials in primate motor cortex encode grasp kinetic parameters," *NeuroImage*, vol. 114, pp. 338–355, Jul. 2015.

[17] A. Khorasani, N. H. Beni, V. Shalchyan, and M. R. Daliri, "Continuous force decoding from local field potentials of the primary motor cortex in freely moving rats," *Sci. Rep.*, vol. 6, no. 1, pp. 1–10, Oct. 2016.

[18] X. She and S. Mukhopadhyay, "SPEED: Spiking neural network with event-driven unsupervised learning and near-real-time inference for event-based vision," *IEEE Sensors J.*, vol. 21, no. 18, pp. 20578–20588, Sep. 2021.

[19] C. Jiang, L. Yang, and Y. Zhang, "A spiking neural network with spike-timing-dependent plasticity for surface roughness analysis," *IEEE Sensors J.*, vol. 22, no. 1, pp. 438–445, Jan. 2022.

[20] S. Yang, H. Wang, Y. Pang, Y. Jin, and B. Linares-Barranco, "Integrating visual perception with decision making in neuromorphic fault-tolerant quadruplet-spike learning framework," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 54, no. 3, pp. 1502–1514, Mar. 2024.

[21] S. Yang, H. Wang, Y. Pang, M. R. Azghadi, and B. Linares-Barranco, "NADOL: Neuromorphic architecture for spike-driven online learning by dendrites," *IEEE Trans. Biomed. Circuits Syst.*, vol. 18, no. 1, pp. 186–199, Feb. 2024.
[22] S. Yang, J. Tan, T. Lei, and B. Linares-Barranco, "Smart traffic navigation system for fault-tolerant edge computing of Internet of Vehicle in intelligent transportation gateway," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 13011–13022, Nov. 2023.
[23] S. Yang, J. Wang, B. Deng, M. R. Azghadi, and B. Linares-Barranco, "Neuromorphic context-dependent learning framework with fault-tolerant spike routing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7126–7140, Dec. 2022.
[24] S. Yang et al., "Scalable digital neuromorphic architecture for large-scale biophysically meaningful neural network with multi-compartment neurons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 148–162, Jan. 2020.
[25] S. Yang et al., "BiCoSS: Toward large-scale cognition brain with multigranular neuromorphic architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2801–2815, Jul. 2022.
[26] S. Yang et al., "Real-time neuromorphic system for large-scale conductance-based spiking neural networks," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2490–2503, Jul. 2019.
[27] J. Liao et al., "An energy-efficient spiking neural network for finger velocity decoding for implantable brain-machine interface," in *Proc. IEEE 4th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2022, pp. 134–137.
[28] B. Zhou, P.-S. V. Sun, and A. Basu, "ANN vs SNN: A case study for neural decoding in implantable brain-machine interfaces," 2023, *arXiv:2312.15889*.
[29] J. Dethier, P. Nuyujukian, S. I. Ryu, K. V. Shenoy, and K. Boahen, "Design and validation of a real-time spiking-neural-network decoder for brain–machine interfaces," *J. Neural Eng.*, vol. 10, no. 3, Apr. 2013, Art. no. 036008.
[30] G. Leone, L. Raffo, and P. Meloni, "ZyON: Enabling spike sorting on APSoC-based signal processors for high-density microelectrode arrays," *IEEE Access*, vol. 8, pp. 218145–218160, 2020.
[31] J. Cheslet et al., "FPGA implementation of a spiking neural network for real-time action potential and burst detection," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2023, pp. 1–5.
[32] E. A. Vallicelli et al., "Real-time digital implementation of a principal component analysis algorithm for neurons spike detection," in *Proc. Int. Conf. IC Design Technol. (ICICDT)*, Jun. 2018, pp. 33–36.
[33] Z. Zhang and T. G. Constandinou, "Firing-rate-modulated spike detection and neural decoding co-design," *J. Neural Eng.*, vol. 20, no. 3, May 2023, Art. no. 036003.
[34] H. An et al., "A power-efficient brain-machine interface system with a sub-mw feature extraction and decoding ASIC demonstrated in non-human primates," *IEEE Trans. Biomed. Circuits Syst.*, vol. 16, no. 3, pp. 395–408, Jun. 2022.
[35] F. Boi et al., "A bidirectional brain-machine interface featuring a neuromorphic hardware decoder," *Frontiers Neurosci.*, vol. 10, p. 563, Dec. 2016.
[36] J. Liao et al., "A spiking neural network decoder for implantable brain machine interfaces and its sparsity-aware deployment on RISC-V microcontrollers," 2024, *arXiv:2405.02146*.
[37] G. Leone, L. Raffo, and P. Meloni, "On-FPGA spiking neural networks for end-to-end neural decoding," *IEEE Access*, vol. 11, pp. 41387–41399, 2023.
[38] T. Brochier et al., "Massively parallel recordings in macaque motor cortex during an instructed delayed reach-to-grasp task," *Sci. Data*, vol. 5, no. 1, pp. 1–23, Apr. 2018.
[39] *Utah Array*. Accessed: Jun. 28, 2024. [Online]. Available: https://blackrockneurotech.com/products/utah-array/#:~:text=What%20is%20the%20Utah%20Array,degree%20of%20precision%20and%20accuracy
[40] J. K. Eshraghian et al., "Training spiking neural networks using lessons from deep learning," *Proc. IEEE*, vol. 111, no. 9, pp. 1016–1054, Sep. 2023.
[41] J. Yik et al., "NeuroBench: A framework for benchmarking neuromorphic computing algorithms and systems," 2024, *arXiv:2304.04640*.

Luca Martis received the B.S. and M.S. degrees in electronics engineering from the University of Cagliari, Cagliari, Italy, in 2021 and 2023, respectively, where he is currently pursuing the Ph.D. degree in electronic and computer engineering.

His research interests include the development of custom hardware systems for executing algorithms based on artificial intelligence, with a particular focus on spiking neural networks.

Gianluca Leone received the B.S. degree in electronics engineering from the University of Cagliari, Cagliari, Italy, in 2016, the M.S. degree in electronics engineering from Politecnico di Torino, Turin, Italy, in 2019, and the Ph.D. degree in electronics and computer engineering from the University of Cagliari, in 2023.

Since 2023, he has been an Assistant Professor with the University of Cagliari, where he teaches integrated systems design and mixed-signal circuits and systems. His recent work includes the development of FPGA-based systems for processing biosignals and SNN-type workloads in real time at the edge. His research interests include the design and optimization of digital systems.

Luigi Raffo (Member, IEEE) received the Laurea degree in electronic engineering and the Ph.D. degree in electronics and computer science from the University of Genoa, Genoa, Italy, in 1989 and 1994, respectively.

In 1994, he joined the Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy, as an Assistant Professor and an Associate Professor in 1998, where he has been a Full Professor of electronics with the Department of Electrical and Electronic Engineering, University of Cagliari, since 2006. He teaches courses on system/digital and analog electronic design and processor architectures for the courses of studies in electronic and biomedical engineering. He was a Coordinator of the project EU IST-FET-IST-2001-39266-BEST and the MADNESS EU Project (FP7/2007-2013), a Unit Coordinator of the project EU IST-FETSHAPES-Scalable Software Hardware Architecture Platform for Embedded Systems, and a Local Coordinator of industrial projects in the field (among others: ST-Microelectronics-Extension of ST200 architecture for ARM binary compatibility and ST-Microelectronics-Network on Chip). He is responsible for the cooperation programs in the field of embedded systems with several other European Universities. He was also a Local Coordinator of the ASAM (ARTEMIS-JU) and ALBA projects (national founded project) and RPCT (regional founded project).

Paolo Meloni (Member, IEEE) is currently an Associate Professor with the University of Cagliari, Cagliari, Italy, where he teaches microcontroller-based systems and advanced embedded systems. He is the author of a significant track of international research papers. His research activity is on the development of advanced digital systems on the application-driven design and programming of multicore on-chip architectures and FPGAs.

Open Access funding provided by 'Università degli Studi di Cagliari' within the CRUI CARE Agreement