Enabling SNN-Based Near-MEA Neural Decoding with Channel Selection: An Open-HW Approach

Gianluca Leone, Luca Martis, Luigi Raffo, Paolo Meloni
Dipartimento di Ingegneria Elettrica ed Elettronica, University of Cagliari, Italy
gianluca.leone94@unica.it, luca.martis@unica.it, raffo@unica.it, paolo.meloni@unica.it

Abstract—**Advancements in CMOS microelectrode array sensors have significantly improved sensing area and resolution, paving the way to accurate Brain-Machine Interfaces (BMIs). However, near-sensor neural decoding on implantable computing devices is still an open problem. A promising solution is provided by Spiking Neural Networks (SNNs), which leverage event sparsity to improve energy consumption. However, given the typical data rates involved, the workload related to I/O acquisition and spike encoding is dominant and limits the benefits achievable with event-based processing. In this work, we present two power-efficient implementations, on FPGA and ASIC, of a dedicated processor for the decoding of intracortical action potentials from primary motor cortex. The processor leverages lightweight sparse SNNs to achieve state-of-the-art accuracy. To limit the impact of I/O transfers on energy efficiency, we introduced a channel selection scheme that reduced bandwidth requirements by 3x and power consumption by 2.3x and 1.6x on the FPGA and ASIC, respectively, enabling inference at 0.446 µJ and 1.04 µJ, with no significant loss in accuracy. To promote broad adoption in a specialized, research-intensive domain, we have based our implementations on open-source EDA tools, low-cost hardware, and an open PDK.**

I. Introduction

Intracortical Brain-Computer Interfaces (iBCIs) establish a direct line of communication between the brain and external devices, allowing the interpretation of the activity of the brain and the retrieval of useful information. This process is known as neural decoding. Example applications swipe from controlling a computer cursor for patients diagnosed with major movement impairments [1], to classifying thoughts into words for individuals who have lost the ability to articulate speech [2].

A new generation of iBCIs is on the way, taking advantage of new CMOS intracortical microelectrode arrays (MEA), sampling neural activity with unprecedented spatial and temporal resolution [3]–[5], and advances in artificial intelligence (AI), where artificial neural networks (ANN) proved to be capable of accurate decoding [6], [7]. However, in practice efficient deployment of iBCIs is still an open research problem. Neural decoding has to operate in real time at the edge, under stringent power and area constraints, as near-sensor execution can improve responsiveness in the case of closed-loop experiments [8] and optimize the energy consumed for data transmission. This requirement is in contrast with the computation and communication workload imposed by high channel counts and the sampling rate of modern MEAs and by the complexity of ANNs used for decoding [9], [10].

This work was supported in part by Key Digital Technologies Joint Undertaking (KDT JU) in EdgeAI "Edge AI Technologies for Optimised Performance Embedded Processing" project, grant agreement No 101097300, 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.

Thus, adequate workload optimization techniques and lightweight, energy-efficient decoding algorithms need to be studied and assessed. Spiking Neural Networks (SNNs) emerge in this scope as accurate and energy-efficient models, thanks to their capability to extract time-encoded features in the data while using an event-based computing paradigm, avoiding unnecessary computations. SNNs have already been utilized in the field of real-time battery-operated iBCIs, providing a lower memory footprint and power consumption in neural decoding tasks compared to alternatives [11]. Moreover, as in several multichannel biosignal processing cases, channel selection can be used to optimize workload. A calibration process is used to assess which channels effectively contribute to neural data interpretation accuracy and to identify which data need to be eventually loaded by the decoder and processed in real time at the edge.

In this work we combine SNN with channel selection, to take a step further in the optimization of neural decoding systems. Moreover, most SNN-based neural decoders are implemented using highly customized proprietary integrated circuits. In contrast, we assess our approach on easily acquired hardware and easily repeatable development processes, focusing on low-cost FPGA devices and on a hardware macro implemented with open EDA tools and an open PDK.

Summarizing, the major novelty presented in this paper is:

- We present a neural decoding methodology that combines lightweight SNN and channel selection, to reduce I/O-related and processing-related power;
- We adapt to our methodology the open-source SNN-processor presented in [12] and assess our approach on the native low-cost and low-power Lattice iCE40UP5K FPGA;
- We adapt the SNN decoder to be implemented as ASIC macro, using an open-hardware approach, exploiting the open-source silicon implementation flow Openlane [13], and the open PDK SKY130[1].

[1]https://skywater-pdk.readthedocs.io/en/main/

II. Related Work

A substantial body of literature explores neural decoding, although most studies do not prioritize hardware implementation. Nevertheless, these works have demonstrated the effectiveness of deep learning techniques, particularly Recurrent Neural Networks (RNNs) such as Quasi-Recurrent Neural Networks (QRNNs) [7] and Long Short-Term Memory (LSTM) networks [6], [7], in decoding hand movements from brain activity. Other works have primarily focused on the initial stages of processing, leaving the decoding algorithm to be executed off-chip [14], [15]. More recent studies have incorporated in-place classification. Some works focus on fairly simple feature extraction and decoding algorithms. For example, [16] uses an ASIC to extract the Spiking Band Power (SBP) — the averaged intracortical signal in the 300 – 1,000 Hz frequency band — and a steady-state Kalman filter (SSKF) as a decoder.

Other works, more related to ours, highlight the interest, in this context, for SNNs, as energy-effective class of decoding algorithms. A seminal article in this area is presented in [8], where a dedicated circuitry (ASIC) implements an SNN-based decoder for a decoding task integrated into a closed-loop experiment. Despite the advanced application case, the decoder does not integrate the whole chain, as it offloads feature extraction to an FPGA. In [17], the authors employed microcontrollers (MCUs) as the target platform. Relying on a quite advanced technology node and on a mature and widely available target, this approach achieves favorable power efficiency and is more widely adoptable for the community than custom ASICs. However, it sacrifices execution time due to the limitations of general-purpose processors when it comes to SNN inference. None of the previously mentioned works focus on the optimization of the I/O-related aspects of the application.

Our work closes the gaps mentioned above. We combine, on the same on-chip processing system, read-in of the sensor data, in-place feature extraction, and decoding. Moreover, we apply channel selection, a widely used workload optimization in multichannel biosignal processing [18], [19], to multi-unit activity (MUA) — intracortical action potentials extracted from raw neural data — and estimate the benefits on hardware.

Moreover, to the best of our knowledge, this paper presents the first neural decoding system starting from an open HDL description of an SNN processor [12], prototyped with low-cost devices, open source EDA tools and an open PDK.

III. Methods

A. Neural decoding scheme

We envision the system to acquire raw signals from an implanted front-end counterpart, as shown in Figure 1. The raw samples undergo a feature extraction process aimed at refining and compressing the neural data by extracting intracortical action potentials (MUA). The spikes can be directly processed in real-time by the SNN-based neural decoder, which is intrinsically compatible with MUA, reacting to event trains.

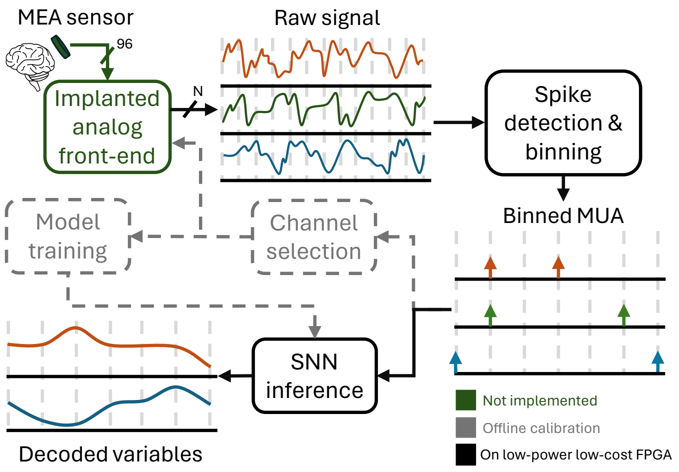

Fig. 1: System overview

During the offline training and calibration phases, a subset of N channels is selected to be used in real time. Channel selection enables reduced power consumption, both in the implanted recording system and in the external neural decoder.
Finally, neural decoding is accomplished by executing the SNN inference. The network is trained on a dataset connecting MUA recordings to the corresponding variables to decode (in this case the position of a cursor and the displacement of an object). The neurons in the output layer of the SNN are set so that their potential follows the evolution of the behavioral data.

B. Reference behavioral task

We used two public datasets containing neural signals acquired from Rhesus macaque monkeys employing chronically implanted 96-channel MEA in the primary motor cortex.
Dataset 1 (indy_20170127_03) [20] is recorded at 24.4 kHz (downsampled at 12.2 kHz in this study) while the subject performs a continuous series of reaches to targets. The behavioral data were presented as the cursor position on the x and y axes, sampled at 250 Hz (upsampled at 1.02 kHz in this study).
Dataset 2 (monkey N) [21] is recorded at 30 kHz (downsampled at 10 kHz in this study) while the subject performs delayed reach-to-grasp tasks. The behavioral data were presented as the grasped handle displacement and the force applied from the subject's fingers, sampled at 1 kHz.

C. Target hardware architecture

We used SYNtzulu [12], an open source SNN processing system[2], as the base platform for the target hardware architecture, shown in Figure 2. We then adapted it to meet the specific requirements of neural decoding. It includes:

- a multiplier-less spike detection and binning pipeline, useful to extract binned MUA from the raw signal, i.e. number of spikes detected within a time window;
- a compact SNN accelerator, which decodes the behavioral variables - velocity and force - from the binned MUA

[2]https://github.com/gianlucaleone/SYNtzulu

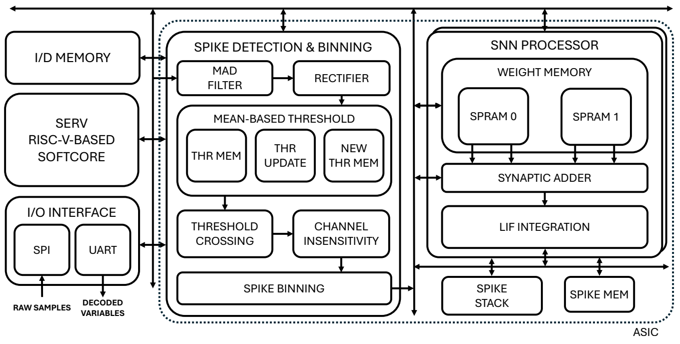

Fig. 2: Hardware architecture

while exploiting the sparsity in the data to speed up the execution and gain idle time;

- a tiny RISC-V softcore (SERV[3]), capable of scheduling I/O data transfer, and of acting as power controller;
- an I/O interface, which streams samples in through a 21 Mbps SPI link and inference out via a 115,200 bps UART.

*1) **Spike detection and binning**:* The spike detection and binning pipeline can be summarized as follows:

- Filtering: a II-order moving average difference (MAD) filter, described by Equation 1, is used to remove low-frequency components.

$$MAD(n) = x(n) - \frac{1}{2}\left[x(n-1) + x(n-2)\right] \tag{1}$$

- Spike emphasis: the filtered signal is rectified. Rectification does not emphasize spike shape, however, this step is fundamental to limit the computational complexity of the threshold evaluation;
- Threshold and detection: the mean value of the rectified signal, updated every 8192 samples, is multiplied by five for Dataset 1, using sum and shift technique, and by four for Dataset 2, and used as spike detection threshold;
- Channel insensitivity: multiple detections are avoided by implementing an insensitivity period after spike detection of 1.2 ms for Dataset 1 and 1 ms for Dataset 2;
- Spike bins: spikes are counted in time intervals of 1.2 ms for Dataset 1 and 1 ms for Dataset 2.

With this configuration, the output frequency of the binned MUA is set to approximately 1 kHz, matching the sampling frequency of the behavioral variables. Moreover, since the insensitivity period equals the binning resolution, the bins can only be one or zero and can be directly processed by the SNN processor, generating an output inference for each input bin.

*2) **SNN processor**:* The dual-core SNN processor supports inference of feedforward SNNs composed of fully connected layers of Leaky Integrate-and-Fire (LIF) neurons. To this end, the SNN processor integrates a dedicated LIF integration module per core that computes the neuron dynamics.
Every core embeds a 4-way synaptic adder to accumulate synaptic weights and compute synaptic current. This task corresponds to the most critical part of the workload in SNN inference in this scope.

[3]SERV: award-winning bit-serial RISC-V core https://github.com/olofk/serv

A fundamental prerogative of SNN algorithms is to be able to exploit the sparsity in the data and between layers, avoiding all unnecessary weight loads and accumulations when synaptic connections are inactive. To this aim, a spike stack tracks active synapses layer by layer, annotating the addresses of active synapses only, and scheduling their accumulation. To reduce the impact of the spike stack instance, synapses are gathered in groups of four, reducing the memory requirements for implementing the stack from 1.75 to 0.31 kb. On the flip side, this reduced the sparsity exploitable by the system from 85% to 58% on average, or to 68%, when neurons are reordered to group more active neurons.

D. Channel Selection Mechanism

Channel selection was performed by ranking the input channels according to the correlation between the firing rate of the channels and the behavioral variables, calculated using equation 2.

$$CC(A,B) = \frac{1}{N-1}\sum_{i=1}^{N} \frac{A_i - \mu_A}{\sigma_A}\frac{B_i - \mu_B}{\sigma_B} \tag{2}$$

Where, CC is the correlation coefficient, N is the number of samples, which corresponds to half of the entries of the training set for this measure, A is the firing rate of an input channel, and B is a behavioral variable.
We selected different subsets of channels among the highest ranked. The removal of channels enables the system to process a reduced amount of input data, thus reducing the execution time to complete the workload in each sampling period. The system can use duty cycling accordingly to optimize energy consumption. Detailed results are presented in Section V.

E. SNN training

The networks were trained using snnTorch [22], with binned MUA as input and force and cursor/handle velocity as target outputs. The network topology follows the structure adopted in [11] for neural decoding with lightweight SNN. It consists of four dense layers: the first three layers contain 64, 128, and 64 neurons, respectively, while the number of neurons in the final layer depends on the number of variables to decode. Specifically, this number is 2 neurons for Dataset 1 and 5 neurons for Dataset 2. This configuration ensures that, during training, each behavioral variable is assigned to an output neuron and used in the MSE-based loss computation. As a result, the model learns to predict the behavioral variables directly through its output neurons membrane potential, eliminating the need for decoding the SNN's output.
As summarized in Table I, for training, we set the learning rate at 1e-3, the batch size to 10, and ran for 100 epochs.

DS	Input	L1	L2	L3	L4	Loss	L. Rate	Batch	Epochs
1	$[4 \div 96]$	64	128	64	2	MSE	1e-3	10	100
2					5				

TABLE I: Training hyperparameters

Following the guidelines in [23], the dataset was partitioned into 80% for training, 10% for validation, and 10% for testing. After training, the weights were quantized to 8 bits using appropriately selected scaling factors to optimize memory usage. In particular, no scaling is required between layers, as data forwarding from layer to layer occurs in spike form. Additionally, a 32-bit memory space is allocated for each neuron potential, determined on the basis of monitoring during inference. The quantization process resulted in negligible accuracy loss.

IV. On hardware implementation

A. Low-power FPGA implementation

The baseline [12] is optimized to fit the low power Lattice iCE40UP5K FPGA to improve energy efficiency. The Lattice iCE40UP5K hosts 5,280 LCs embedding both a 4-bit input LUT and a flip-flop (FF), 30 4-Kb BRAM tiles, 4 256-Kb SPRAMs, and 8 DSP slices containing 16x16-bit multipliers. Logic synthesis and implementation were carried out using the open-source framework Yosys + Netxtpnr [24]. In this specific case, we have optimized the baseline implementation to align with the encoding and training scheme as follows:

- The system runs the spike detection process 1.22 times more frequently for Dataset 1, due to its higher sampling rate of 12.2 kHz;
- The binning process accumulates 12 spike times to align the 12.2 kHz neural data sampling rate with the 1.02 kHz behavioral data sampling rate;
- The memory for membrane potentials and the LIF neuron integration modules have been modified to support 32-bit data representation, as required by the SNN training library snnTorch;

The utilization of resources is shown in Table II.

LC	DSP	BRAM	SPRAM
4,528 (85%)	4 (50%)	26 (86%)	4 (100%)

TABLE II: Resource utilization on Lattice iCE40UP5K

LC utilization is 85%, indicating the potential to accommodate a simple module to close the loop. All the embedded SPRAMs are in use to enhance the weight reading throughput, even though less than 16% of the memory is utilized, therefore larger SNNs can be executed if necessary. BRAM utilization is 86%, with BRAM resources distributed throughout the system to save LCs. Finally, 4 DSPs are used to integrate neurons in the *SNN processor*.
In the FPGA implementation, the impact of channel selection on resource usage is minimal, as the SNN accelerator is not involved and the processing elements of the spike detection and binning pipeline remain unchanged - both LCs and DSP units are not affected - and the BRAMs utilized were already operating at only 37.5% capacity (96 channels × 16 bits = 1.5 Kb, with BRAM blocks of 4 Kb).

B. ASIC macro implementation

To assess the benefits of our approach on an ASIC implementation, we have implemented a hardware macro usable in custom SoCs, including all the circuitry related to MUA extraction and decoding as highlighted in Figure 2. The macro, shown in Figure 3, is hardened using OpenLane [13] as EDA suite and SKY130 as a PDK, and is thought to be usable inside compatible harness SoC platforms, such as e.g., Caravel[4]. OpenLane is an RTL-to-GDSII flow including Yosys [24] for synthesis, OpenROAD [25] for floorplanning, placement, and routing, KLayout[5] for GDSII file generation, and Magic[6] and Netgen[7] for Design Rule Checks and Layout Versus Schematic verification. The design was adapted to fit the

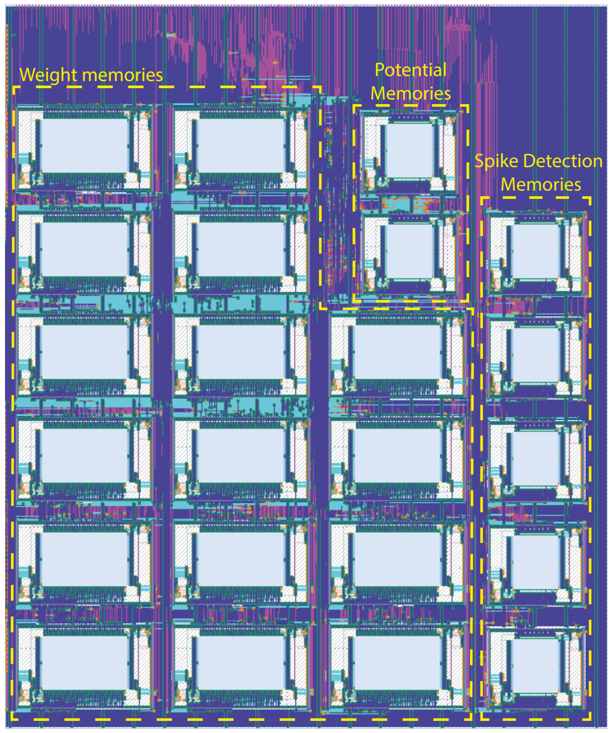

Fig. 3: Design layout generated using the OpenLane flow and SKY130 PDK. On the left and center, the sixteen memories (32 KB) store the weights. At the top right, the two memories (2 KB) are used for storing the membrane potential of the neurons. On the right, the five memories (5 KB) are dedicated to the spike detection modules.

PDK by replacing FPGA-specific devices such as BRAM, SPRAM, and DSP blocks with components available in the PDK, as summarized in Table III. Specifically, the DSPs were replaced with combinational logic, while BRAMs and SPRAMs were substituted with the memories available in the PDK and flip-flops. In particular, weight memories, which were implemented in the FPGA using two SPRAM blocks,

[4]https://caravel-harness.readthedocs.io/en/latest/
[5]https://www.klayout.de/
[6]https://github.com/RTimothyEdwards/magic
[7]https://github.com/RTimothyEdwards/netgen

TABLE III: ASIC macro report

Core Area	3000x3600 μm*
Combinational cells	7311
Sequential cells	1667
SRAM memory	39 KB (23 macros)
F_{max}	100 MHz

*Compatible with Caravel

were implemented in the ASIC using sixteen 2 KB dual-port SRAMs coordinated with additional logic. Other memories were implemented using 1 KB dual-port SRAM memories, except for spike memories. The latter did not fit well with the available memory sizes, making it more area-efficient to implement these parts using flip-flops.

V. Experimental Results

A. Power-Accuracy Trade-off

Table IV reports the validation set accuracy in terms of coefficient of determination R^2 for Dataset 1 and 2 for several subsets of input channels. Specifically, Dataset 1 does not show accuracy loss moving from 96 to 64 recording sites. In the case of Dataset 2, the velocity accuracy stays almost constant from 96 to 16 electrodes. However, the force accuracy shows an appreciable decrement moving from 32 to 16 inputs. These

Channels		96	64	32	16	8	4
DS 1	Vel	0.65	0.65	0.56	0.51	0.44	0.28
DS 2	Vel	0.75	0.76	0.78	0.75	0.73	0.65
	Force	0.84	0.79	0.79	0.69	0.70	0.45

TABLE IV: Decoding accuracy on validation set

results confirm the validity of our channel selection method, as the accuracy achieved using a subset of input channels is often comparable to the one obtained using all recording sites.

Similarly, Table V shows the average power consumption, over a decoding cycle, for different numbers of input channels, considering the sampling rate of 12 kHz for Dataset 1, and 10 kHz for Dataset 2, while the system is clocked at 22.5 MHz in both our target implementation experiments.

Channels	96	64	32	16	8	4
FPGA @ 12 kHz [mW]	11.95	8.50	5.06	3.34	2.48	2.05
FPGA @ 10 kHz [mW]	10.25	7.37	4.50	3.06	2.34	1.98
ASIC @ 12 kHz [mW]	2.89	2.31	1.74	1.45	1.31	1.24
ASIC @ 10 kHz [mW]	2.66	2.16	1.67	1.42	1.30	1.24

TABLE V: Average power consumption

The power consumption of the FPGA-based system was obtained by measuring the voltage drop across three shunt resistors soldered on the power supply lines. The softcore gates the clock of idle modules to save power and implement duty cycling; therefore, channel selection, increasing idle time for all the I/O and spike detection modules, significantly reduces average power. In the baseline 96-channel implementation the I/O contributes to the 29% of the overall power consumption. By applying channel selection, data transfer impact, is lowered to 22% for 64 channels, and to 16% for 32 channels. Finally, peak and idle power are 13.5 and 1.3 mW.

The power consumption of the ASIC macro was estimated through post-synthesis simulation and activity-aware power estimation. Compared to the FPGA, which exploits a more advanced 40 nm technology, our ASIC macro has a higher peak power consumption, around 28 mW. On the other hand, FPGA suffers from a significant quiescent power during idle, while the ASIC macro, when gated, only dissipates static power inside standard cells and memories, thus showing a lower average dissipation even before channel selection.

Figure 4 summarizes the results presented in the previous sections. The plots show the power consumed by the FPGA-based neural decoder and the ASIC SNN decoder on the y-axis, the accuracy of the validation set of the deployed SNNs on the bottom x-axis, and the number of channels on the top x-axis. In the left plot for Dataset 1, the 64-channel

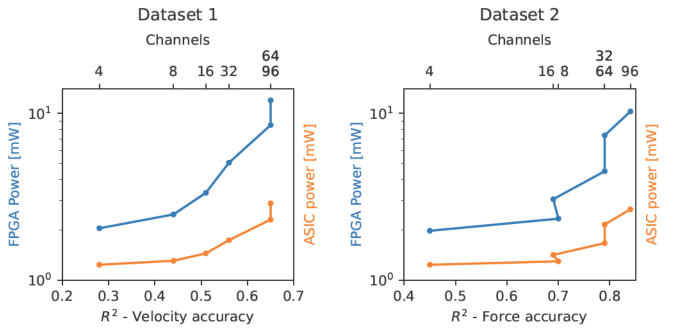

Fig. 4: Power-Accuracy trade-off

implementation achieves the same accuracy as the 96-channel implementation while reducing power consumption by 20% for the ASIC and 29% for the FPGA. In the right plot for Dataset 2, we report the force decoding accuracy, since the velocity accuracy was nearly constant. The 32-channel design outperforms the 64-channel one, achieving the same accuracy while consuming 39% less power for the FPGA and 23% less for the ASIC. Therefore, if the application tolerates a slight reduction in the decoding accuracy without losing the velocity accuracy, 32 channels can be used, allowing a 56% reduction in power consumption.

Figure 5 depicts the decoded velocity in the test set of Dataset 1 for several decoders and the ground truth. The 96, 64, and 32

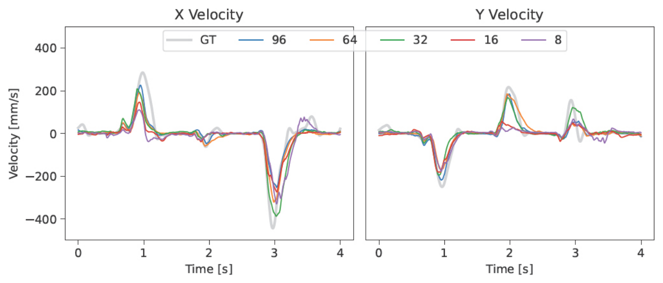

Fig. 5: Decoded velocity (from dataset 1 test set)

input decoders perform similarly, but the quality deteriorates

Work	Year	Channels	Sample rate [kHz]	Feature Extraction	Task	Platform	Process [nm]	Decoder	Power [mw]	Energy/inf. [uJ]	Time/inf. [ms]
our	2024	64	12	MUA	Finger 2D	FPGA	40	SNN	8.50	0.446	0.03
our	2024	64	12	MUA	Finger 2D	ASIC	130	SNN	2.31	1.04	0.03
[16]	2022	93	2	SBP	Finger 2D	ASIC	180	SSKF	0.588	-	-
[17]	2024	96	2	N/A	Finger 2D	MCU	22	SNN	0.5	1.88	0.12
[8]	2016	15	25	N/A	Object control	ASIC	180	SNN	4	-	-

TABLE VI: Hardware implementation comparison

when fewer channels are employed.
In the next section, in the comparison with the state of the art, we focus on the 64-input implementation for dataset 1 and on the 32-input implementation for dataset 2. Therefore, we measured the accuracy of the models on the test set, obtaining $R^2 = 0.73$, and $CC = 0.86$ for dataset 1, and $R^2 = 0.68$, and $CC = 0.85$ for velocity decoding in dataset 2, and $R^2 = 0.78$, and $CC = 0.88$ for force decoding in dataset 2.

VI. Comparison with State of Art

Table VII lists all the works we found in the literature that targeted the same data sets utilized in this study. The first half of Table VII is related to Dataset 1. To the best of our knowledge, among all the models trained to address neural decoding on Dataset 1, no one is deployed on edge-computing platforms, nor tailored for real-time applications. Moreover, the size of the models, expressed in terms of the number of parameters, shows a memory footprint between 7.54 and 16.6 times higher than the solution presented in this work. Finally, the accuracy of the related works is comparable, although, the proposed solution considers only 64 recording sites and relies on real-time spike detection, rather than on intracortical spikes available in the dataset and obtained using offline methods.

DS	Works	Year	Model	Platform	Parameters	CC
1	**This work (64 ch)**	2024	SNN	FPGA	20.6k	0.86
	[7]	2022	QRNN	PC	327k	0.85
	[7]	2022	LSTM	PC	341k	0.84
	[7]	2022	MLP	PC	155k	0.83
2	**This work (32 ch)**	2024	SNN	FPGA	18.8k	0.85
	[11]	2023	SNN	FPGA	22.3k	0.84
	[6]	2021	RNN	PC	420k	0.91

TABLE VII: Accuracy comparison

The second half of Table VII details the works that address neural decoding on Dataset 2. Compared to previous real-time solutions [11], our work demonstrates the highest velocity decoding accuracy and the lowest number of parameters, requiring 1.19 times fewer parameters than [11] while consuming 8% of power. Compared to non-real-time works, our accuracy is slightly lower than [6]. However, the model presented in [6], requiring 22.34 times more parameters than our SNN, is not suited for edge applications and relies on offline computed MUA signal.

In Table VI, we compare our two implementations with systems dedicated to neural decoding. Energy per inference is computed considering the peak active power, thus is higher for the ASIC than for the FPGA. The works in [16] and [17] have lower power consumption than both of our implementations. [16] implements a fairly lightweight feature extraction and decoding algorithm, thus being hardly comparable in terms of computational efficiency. The work in [17] is SNN-based and uses an MCU fabricated with an advanced 22 nm technology node. It shows higher inference time, resulting in higher energy per inference. The work in [8] shows lower power consumption than our FPGA solution but higher than that of our ASIC macro, even without including spike detection and filters. In summary, our approach, although slightly higher in peak power consumption than the alternatives, consumes less energy per inference with respect to [17], the only work in our comparison reporting such metric.

VII. Conclusion

In this study, we presented two power-efficient implementations of a neural decoding system, on FPGA and ASIC, using an open hardware architecture, low-cost target platforms, and open HDL-to-silicon EDA tools. In the context of iBCIs, we have utilized lightweight SNNs to enhance computational efficiency by exploiting the sparsity of intracortical neural data. Moreover, we assessed a channel selection scheme based on the linear correlation between the channels' firing rates and the target output, reducing bandwidth requirements by up to 3x. This approach enabled power consumption of 8.5 mW on the FPGA and 2.31 mW on the ASIC macro, with energy per inference of 0.446 µJ and 1.04 µJ, respectively, paving the way for advancements in battery-operated iBCIs.

References

[1] C. Pandarinath, P. Nuyujukian *et al.*, "High performance communication by people with paralysis using an intracortical brain-computer interface," *Elife*, vol. 6, Feb. 2017.

[2] D. A. Moses, S. L. Metzger *et al.*, "Neuroprosthesis for decoding speech in a paralyzed person with anarthria," *New England Journal of Medicine*, vol. 385, no. 3, pp. 217–227, 2021.

[3] F. Boi, N. Perentos *et al.*, "Multi-shanks sinaps active pixel sensor cmos probe: 1024 simultaneously recording channels for high-density intracortical brain mapping," *BioRxiv*, p. 749911, 2019.

[4] G. N. Angotzi, F. Boi *et al.*, "Sinaps: An implantable active pixel sensor cmos-probe for simultaneous large-scale neural recordings," *Biosensors and Bioelectronics*, vol. 126, pp. 355–364, 2019.

[5] N. A. Steinmetz, C. Aydin *et al.*, "Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings," *Science*, vol. 372, no. 6539, p. eabf4588, 2021.

[6] S.-H. Yang, J.-W. Huang *et al.*, "Selection of essential neural activity timesteps for intracortical brain–computer interface based on recurrent neural network," *Sensors*, vol. 21, no. 19, p. 6372, 2021.

[7] N. Ahmadi, T. Adiono *et al.*, "Improved spike-based brain-machine interface using bayesian adaptive kernel smoother and deep learning," *IEEE Access*, vol. 10, pp. 29 341–29 356, 2022.

[8] F. Boi, T. Moraitis *et al.*, "A bidirectional brain-machine interface featuring a neuromorphic hardware decoder," *Frontiers in neuroscience*, vol. 10, p. 563, 2016.

[9] S.-J. Kim, S.-H. Han *et al.*, "A sub-μw/ch analog front-end for δ-neural recording with spike-driven data compression," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, pp. 1–14, 2019.

[10] D. Valencia, P. P. Mercier, and A. Alimohammad, “Efficient in vivo neural signal compression using an autoencoder-based neural network,” *IEEE Transactions on Biomedical Circuits and Systems*, 2024.

[11] G. Leone, L. Martis *et al.*, “Spiking neural networks for integrated reach-to-grasp decoding on fpgas,” in *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2023, pp. 1–5.

[12] G. Leone, M. A. Scrugli *et al.*, “Syntzulu: A tiny risc-v-controlled snn processor for real-time sensor data analysis on low-power fpgas,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–12, 2024.

[13] M. Shalan and T. Edwards, “Building openlane: A 130nm openroad-based tapeout- proven flow : Invited paper,” in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–6.

[14] A. J. Bullard, S. R. Nason *et al.*, “Design and testing of a 96-channel neural interface module for the networked neuroprosthesis system,” *Bioelectronic Medicine*, vol. 5, pp. 1–14, 2019.

[15] J. Lim, E. Moon *et al.*, “26.9 a $0.19\times$ 0.17 mm 2 wireless neural recording ic for motor prediction with near-infrared-based power and data telemetry,” in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 416–418.

[16] H. An, S. R. Nason-Tomaszewski *et al.*, “A power-efficient brain-machine interface system with a sub-mw feature extraction and decoding asic demonstrated in nonhuman primates,” *IEEE transactions on biomedical circuits and systems*, vol. 16, no. 3, pp. 395–408, 2022.

[17] J. Liao, O. Toomey *et al.*, “A spiking neural network decoder for implantable brain machine interfaces and its sparsity-aware deployment on risc-v microcontrollers,” *arXiv preprint arXiv:2405.02146*, 2024.

[18] X. Wang, M. Hersche *et al.*, “Mi-bminet: An efficient convolutional neural network for motor imagery brain–machine interfaces with eeg channel selection,” *IEEE Sensors Journal*, vol. 24, no. 6, pp. 8835–8847, 2024.

[19] D. Valencia, G. Leone *et al.*, “Power-efficient in vivo brain-machine interfaces via brain-state estimation,” *Journal of neural engineering*, vol. 20, no. 1, p. 016032, 2023.

[20] J. E. O’Doherty, M. M. B. Cardoso *et al.*, “Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology,” *Zenodo*, may 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3854034

[21] T. Brochier, L. Zehl *et al.*, “Massively parallel recordings in macaque motor cortex during an instructed delayed reach-to-grasp task,” *Scientific data*, vol. 5, no. 1, pp. 1–23, 2018.

[22] J. K. Eshraghian, M. Ward *et al.*, “Training spiking neural networks using lessons from deep learning,” *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023.

[23] J. Yik, K. Van den Berghe *et al.*, “Neurobench: A framework for benchmarking neuromorphic computing algorithms and systems,” *arXiv*, 2023.

[24] D. Shah, E. Hung *et al.*, “Yosys+ nextpnr: an open source framework from verilog to bitstream for commercial fpgas,” in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2019, pp. 1–4.

[25] D. B. T. Ajayi, “Openroad: Toward a self-driving, open-source digital layout implementation tool chain,” *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, 2019. [Online]. Available: https://par.nsf.gov/biblio/10171024