TU Delft

Delft University of Technology

Adaptive Robotic Arm Control with a Spiking Recurrent Neural Network on a Digital Accelerator

Linares-Barranco, Alejandro; Prono, Luciano; Lengenstein, Robert; Indiveri, Giacomo; Frenkel, Charlotte

DOI
10.1109/ICECS61496.2024.10849226

Publication date
2025

Document Version
Final published version

Published in
2024 31st IEEE International Conference on Electronics, Circuits and Systems, ICECS 2024

Citation (APA)
Linares-Barranco, A., Prono, L., Lengenstein, R., Indiveri, G., & Frenkel, C. (2025). Adaptive Robotic Arm Control with a Spiking Recurrent Neural Network on a Digital Accelerator. In *2024 31st IEEE International Conference on Electronics, Circuits and Systems, ICECS 2024* (Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems). IEEE. https://doi.org/10.1109/ICECS61496.2024.10849226

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Adaptive Robotic Arm Control with a Spiking Recurrent Neural Network on a Digital Accelerator

Alejandro Linares-Barranco[1], Luciano Prono[2], Robert Lengenstein[3], Giacomo Indiveri[4], Charlotte Frenkel[5]
[1]*Robotics and Computer's Tech Lab. University of Sevilla*. Sevilla, Spain. (alinares@atc.us.es)
[2]*Dpto. of Electronics and Telecom. Politecnico di Torino*. Torino, Italy
[3]*Institute of Theoretical Computer Science. Graz University of Technology*. Graz, Austria
[4]*Institute of Neuroinformatics. University of Zurich*. Zurich, Switzerland
[5]*Dept of Microelectronics. Delft University of Technology*. Delft, The Netherlands

2024 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS) | 979-8-3503-7720-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICECS61496.2024.10849226

***Abstract*—With the rise of artificial intelligence, neural network simulations of biological neuron models are being explored to reduce the footprint of learning and inference in resource-constrained task scenarios. A mainstream type of such networks are spiking neural networks (SNNs) based on simplified *Integrate and Fire* models for which several hardware accelerators have emerged. Among them, the "ReckOn" chip was introduced as a recurrent SNN allowing for both online training and execution of tasks based on arbitrary sensory modalities, demonstrated for vision, audition, and navigation. As a fully digital and open-source chip, we adapted ReckOn to be implemented on a Xilinx Multiprocessor System on Chip system (MPSoC), facilitating its deployment in embedded systems and increasing the setup flexibility. We present an overview of the system, and a Python framework to use it on a Pynq ZU platform. We validate the architecture and implementation in the new scenario of robotic arm control, and show how the simulated accuracy is preserved with a peak performance of 3.8M events processed per second.**

***Index Terms*—Recurrent SNN, online learning, neuromorphic engineering, FPGA, MPSoC, Python.**

I. Introduction

Spiking neural networks (SNNs) are a particular variant of neural networks, which use pulse-based signals to process information. Unlike traditional artificial neural networks (ANNs), which compute the weighted sum of inputs through multiply-accumulate operations [1], SNNs rely on accumulation operations triggered by binary spikes [2]. This property allows for energy efficiency advantages in sparse scenarios [3]. The implementation of SNN accelerators in hardware aims to exploit this efficiency advantage through highly parallel architectures. It is a research area that has attracted much attention in recent years and utilized both analog and digital design techniques, both synchronous and asynchronous, based on application-specific integrated circuits (ASICs) [4]–[9] or field-programmable-gate-arrays (FPGAs) [10], [11].The state of art has shown that these SNN accelerators have great potential for use in a wide range of applications, such as speech and gesture recognition, or robot navigation and control [9], [12], [13]. These applications require real-time data processing and high energy efficiency, making SNN accelerators an ideal solution.

This research was partially supported by the Spanish Ministry for Digital Transformation and Public Function through grant USECHIP (TSI-069100-2023-001) of PERTE Chip Chair program, funded by European Union – Next Generation EU, by NEKOR (PID2023-149071NB-C54/AEI/10.13039/501100011033) and by the SMALL (PCI2019-111841-2/AEI/10.1309/501100011033) EU CHIST-ERA.

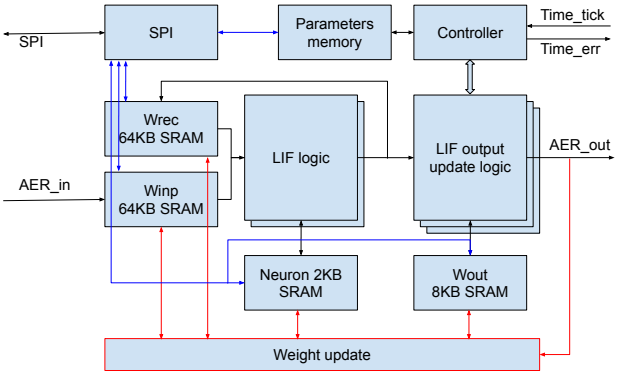

Fig. 1. Block diagram of the ReckOn accelerator (simplified from [9]).

In this paper, we demonstrate an SNN used to perform online adaptive robotic arm control using an open-source recurrent SNN (RSNN) accelerator denoted as "ReckOn" [9], which is implemented in the Verilog hardware description language. Our contributions are two-fold: (i) we integrate ReckOn as part of a full system deployed on a Xilinx multiprocessor programmable system (MPSoC), the Zynq UltraScale+ (ZU), to seamlessly configure and interact with the hardware accelerator through an online Python interface running on an embedded Jupyter server; (ii) using a Pynq-ZU board, we demonstrate the full system in the real-world task scenario of adaptive robotic arm control, exploiting the ability of ReckOn to learn online.

This paper is organized as follows. Section II first describes the ReckOn accelerator in brief. Section III then explains the details and additional circuits needed for the system integration. Finally, Section IV presents the results, after which we summarize the main outcomes.

II. The RSNN accelerator ReckOn

The ability to learn short- and long-term temporal dependencies in embedded hardware is one of the key missing elements needed to improve the robustness of autonomous devices in the real-world, guarantee user privacy, and reduce reliance on

Authorized licensed use limited to: TU Delft Library. Downloaded on February 26,2025 at 10:11:52 UTC from IEEE Xplore. Restrictions apply.

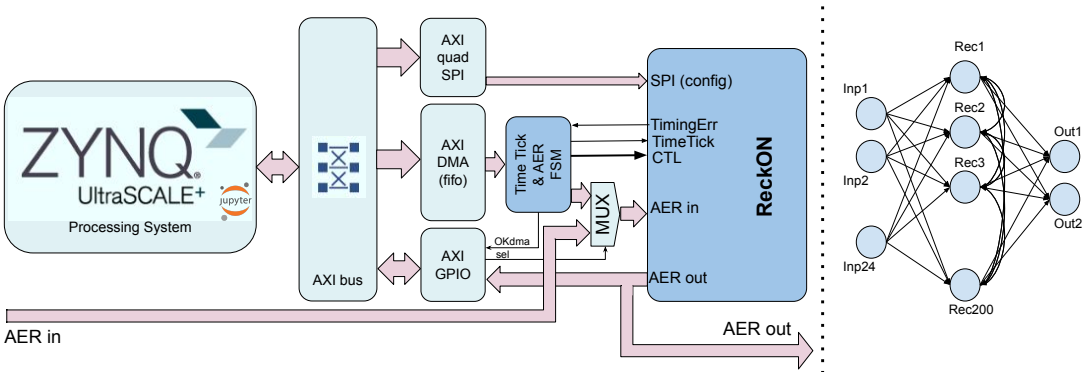

Fig. 2. Left: Block diagram of ReckOn on the Pynq-ZU board. Dark blue blocks are implemented in the PL to support ReckON, while light blue blocks correspond to the PS and IPs from Xilinx library for connectivity. Right: RSNN used in the experiments.

the cloud [9]. To address this challenge, ReckOn proposes an RSNN processor that enables supervised learning over seconds while keeping a millisecond-range temporal resolution. As the vanilla backpropagation-through-time (BPTT) training algorithm requires backpropagating error information through the network dynamics, its memory requirements do not fit resource-constrained autonomous devices for learning over long timescales. To solve this challenge, Bellec et al. proposed the eligibility propagation (e-prop) algorithm [14], which is a forward-mode learning algorithm based on eligibility traces (ETs) that provides a bio-plausible alternative that approximates BPTT. ReckOn implements a modified version of e-prop that is fully local in both space and time, thereby drastically reducing memory requirements by only scaling with the number of neurons, instead of the number of synapses [9].

Figure 1 shows a simplified diagram of the ReckOn accelerator architecture. The RSNN architecture is based on a layer of 256 recurrently connected leaky integrate-and-fire (LIF) neurons, to which 256 input neurons and from which 16 output readout neurons are fully connected. The corresponding recurrent, input, and output weight matrices (W_{rec}, W_{inp}, W_{out}), containing 8-bit weights, as well as the neuron state ($NeuronSRAM$) are stored in on-chip SRAM.

Typically, temporal multiplexing and asynchronous communication techniques are used to transmit the spikes from source neurons to destination ones from one chip to another. The most common one is the Address-Event-Representation (AER) [15]. ReckOn has a spiking AER input bus to allow interfacing with arbitrary neuromorphic sensors, such as the retina [16] and cochlea [17], and an serial-peripheral-interconnect (SPI) bus for initial configuration and debugging. The optional $Time\_tick$ signal allows synchronizing the internal neural processing steps with an external timer, which indicates the temporal resolution of the input data. The module ($Weight\_update$) carries out updates based on the modified e-prop algorithm. ReckOn also exploits sparsity on input data and weight updates to reduce the energy footprint. Prototyped in a 28-nm CMOS node, ReckOn was demonstrated for real-time task-agnostic learning of gesture recognition, keyword spotting, and navigation tasks within power budgets not exceeding 50 μW [9]. Implemented in the Verilog hardware description language, it is available in open source [18].

III. MPSoC Implementation

We deployed ReckOn on a Xilinx MPSoC, model Zynq Ultrascale+ XCZU5EG, for the Pynq-ZU board [19]. This chip from Xilinx consists of an embedded processing system (PS) based on four 64-bit ARM Cortex A53 processors, two ARM Cortex R5F real-time processors, an ARM MALI 400MP GPU-type graphics processor, along with an FPGA (programmable logic, PL) on which to deploy the desired circuitry and communicate with the PS. PS and PL are connected through the Advanced-eXtensible-Interconnect (AXI) bus with direct-memory-access (DMA) support. Based on a Petalinux operating system, we deploy a Jupyter Notebook server in the PS, which allows driving the FPGA using Python through Pynq libraries. We use it to program the PL, to preprocess data to streamline the PS-PL communication, and to collect and analyze the output from ReckOn. The Xilinx Vivado tool, version 2021.2, has been used for development. The Jupyter Notebook server running on the Pynq-ZU board can be operated from a computer, through USB connection and also allows uploading to the PL the bitstream files generated from Vivado.

Fig. 2 (left panel) shows the integration of ReckOn in the Xilinx MPSoc system, resulting from five key design decisions, as follows.

- ReckOn's input AER interface can be chosen to come either (i) from an actual AER sensor interfaced with the board, or (ii) from the PS through an AXI-DMA-interfaced FIFO storing tuples of Address-Events and Timestamps (AE,TS) of samples in a given dataset. This mux-selectable scheme improves flexibility, facilitates debugging and testing of the accelerator, and is easier to deploy with offline datasets.
- To ensure precise timing control within a sample, we use two concatenated finite state machines (FSM) ($Time\_tick$ & $AER\ FSM$ block), deployed on FPGA, instead of using an AXI-GPIO interface, for PS deployment, whose latency would lead to timing distortion. One

Authorized licensed use limited to: TU Delft Library. Downloaded on February 26,2025 at 10:11:52 UTC from IEEE Xplore. Restrictions apply.

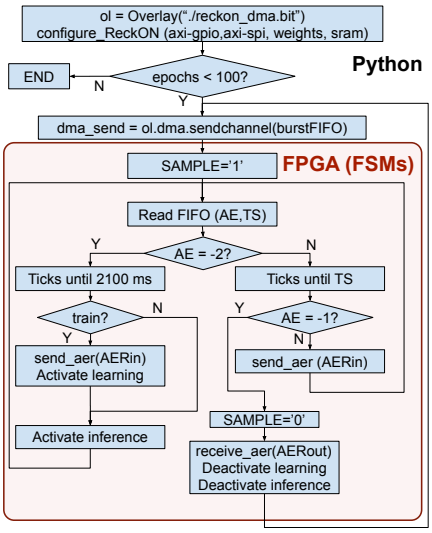

Fig. 3. ReckOn test flow diagram

FSM keeps reading the FIFO containing the events of a dataset sample, while the second controls the sending of these events to ReckOn while ensuring correct timing. More specifically, these FSMs are driven by the PS based on the dataset requirements and take care of the start and end of each sample, of the Req/Ack handshake for each event in a given sample, of providing ticks defining the time reference, of enabling the weight update module for learning with training samples (disabling learning for inference with test samples), and of reading back Reckon's output to provide it to the PS.

- The output of ReckOn, provided at each timestep (regression) or at the end of a sample (classification), also follows an AER format. It can either be sent to the PS through an AXI-GPIO interface, or forwarded to the outside world via the Pynq-ZU board.
- ReckOn requires a configuration interface based on the SPI bus to configure the network, the network weights, and the different configuration parameters available. It is interfaced via a Xilinx AXI-quad-SPI interface that is being used in basic SPI mode.
- As ReckOn is implemented in the PL, the SRAM memories of the original design have been swapped for block RAM (BRAM) resources available inside the FPGA.

IV. Experimental Results

We deploy the MPSoC-based ReckOn system following the flow diagram shown in Fig. 3. A first configuration phase uses the SPI to initialize the RSNN parameters and state, the configuration registers, and the weight memories with random values. Then, the actual training phase takes place and unfolds over a given number of epochs (default: 100). The FSM first triggers the $SAMPLE$ signal to ReckOn to indicate the start of a new sample, which itself contains a sequence of events described as (AE,TS) tuples (Section III), where AE will be written in the address field of the input AER bus, and TS will determine the waiting time, in ticks, until the event is processed. Two special AEs: (i) -2 is used to indicate the classification label (per sample) or regression value (per timestep), which is used as a target during training and as a ground truth during inference to determine the obtained accuracy, and (ii) -1 is used to indicate the end of a sample. Once the end of the sample is reached, the SAMPLE signal is deasserted and the output of ReckOn is retrieved.

The first experiment, implemented for validation, replicates the testbench from [18] for the delayed cue accumulation scenario (also known as *T maze*) of a mobile robot avoiding obstacles as in [9], preserving the accuracies of 100% on the training set and 98% on the test set. To further demonstrate our MPSoC-based ReckOn system, we deploy it in an adaptive robotic arm control use case, where the goal is to determine if there is a weight attached to a robotic arm gripper while it is executing a lemniscate trajectory [20]. The robotic arm gripper is the ED-Scorbot [21], which has four degrees of freedom (DoF). Its motor controllers are spike-based PIDs (SPID) implemented on a Zynq-7100 FPGA, which can receive target joint angles (as spike frequency references) every 100ms to execute any trajectory within its working area. Figure 4 (left panel) shows a block diagram of the scenario. Each SPID (split into ID + P blocks) receives a spiking signal representing the error between the input spiking target angle and the current angle of the joint, also as a spiking signal. The spiking output of each SPID is used to drive the corresponding joint's motor. Each of these spiking signals has polarity. These spiking activities have been recorded while the robot was executing 18 different lemniscate-shape trajectories [20]. Each trajectory was repeated with and without a 1-kg weight attached to the gripper. While maintaining the direction changes of the joints, the recorded spiking activity has been filtered and shrunk to a duration of 2250 ms, corresponding to the sample duration to be learned and classified by ReckOn. We have split the dataset into 50% of the samples for training and the rest for testing.

We used a 24-200-2 network topology on ReckOn, as per Fig. 2 (right panel), with 200 recurrent neurons, 24 input neurons, corresponding to the 6 spiking activity sources from the 4 SPIDs of the dataset, and two output neurons denoting the two possible decisions, i.e. no weight attached to the gripper or 1-kg. In order to determine the best hyperparameters for the time constants, firing thresholds, and learning parameters of this RSNN, we have used the *Weights & Biases* tool [22] to automate the hyperparameter search while ReckOn is running online in the MPSoC. Figure 4 (right panel) shows the test accuracy evolution for 500 executions with different hyperparameters with 100 epochs/execution. The best hyperparameters lead to an accuracy of 88.9% for the train set and of 83.3% on the test set for 100 epochs (Figure 5). The measured peak rate of input events processed via the AER input bus of ReckOn amounts to 3.8M events per second, with an average of 18k events per second from Python experiments.

The resource utilization in the PL amounts to 36% of the available LUTs (30% for ReckOn), 47% of the BRAM resources (30% for ReckOn), and 12% of DSP blocks.

Authorized licensed use limited to: TU Delft Library. Downloaded on February 26,2025 at 10:11:52 UTC from IEEE Xplore. Restrictions apply.

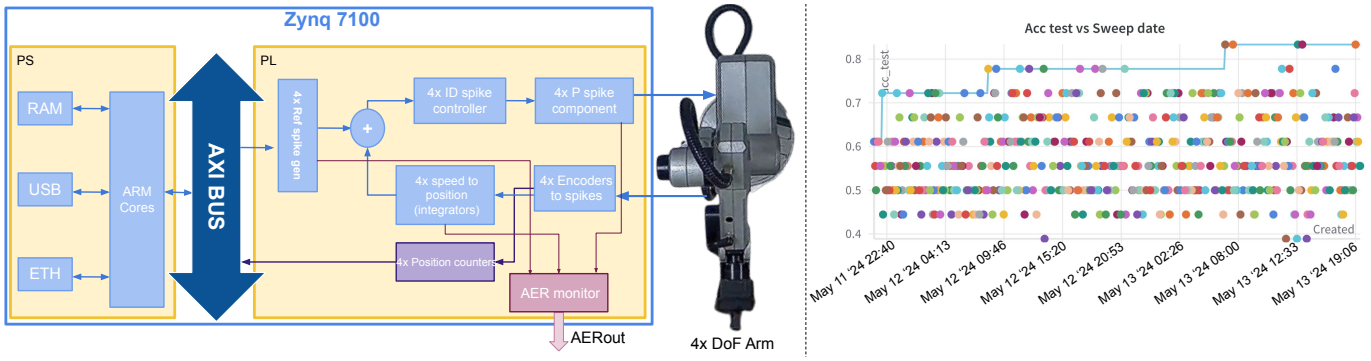

Fig. 4. Left: ED-Scorbot SPID controllers and spiking activity recording scenario for the collected dataset. Right: W&B Accuracy results on tests vs hyperparameter search date-time (dots' colors were randomly assigned by Weight & Bias tool).

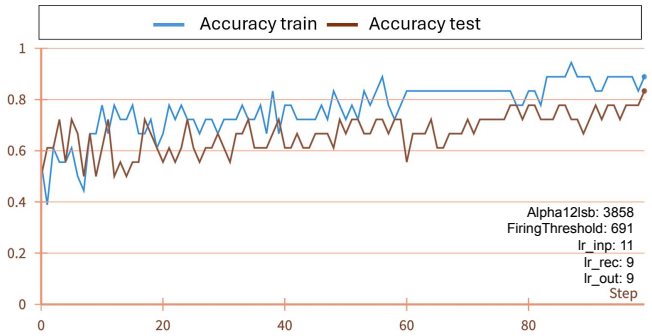

Fig. 5. Accuracy evolution vs 100 epochs/steps for the train/test datasets using the best hyperparameters found by the W&B tool.

V. Conclusions

In this work, the spiking neural network accelerator ReckOn, an open-source digital RSNN that embeds online learning on temporal tasks, has been deployed on a Xilinx MPSoC platform that is part of a Pynq-ZU board. The different techniques and circuits used for this implementation and their control from a Jupyter Notebook are presented. We deployed it for an adaptive robotic arm control use case, where the objective is to detect the weight on a robotic arm gripper, demonstrating the ability of the proposed system to learn complex temporal dependencies in the real world. The required resource utilization in the PL amounts to about 30%. These are promising results for a system that is able to carry out training and inference for an RSNN without any reliance on cloud systems. This MPSoC system therefore directly contributes to low-power, low-latency neuromorphic Edge-AI applications.

References

[1] M. Rabinovich and et al., "Dynamics of sequential decision making," *Physical Review Letters*, vol. 97, 2006.

[2] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[3] P. U. Diehl and et al., "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *IEEE Int. Conference on Rebooting Computing (ICRC)*, 2016, pp. 1–8.

[4] M. Khan and et al., "Spinnaker: Mapping neural networks onto a massively-parallel chip multiprocessor," in *2008 IEEE International Joint Conference on Neural Networks*, 2008, pp. 2849–2856.

[5] J. Sawada and et al., "TrueNorth ecosystem for brain-inspired computing: scalable systems, software, and applications," in *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2016, pp. 130–141.

[6] S. Moradi and et al., "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[7] B. V. Benjamin and et al., "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

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

[9] C. Frenkel and G. Indiveri, "Reckon: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[10] K. Cheung and et al., "A large-scale spiking neural network accelerator for fpga systems," in *Artificial Neural Networks and Machine Learning – ICANN 2012*. Springer Berlin Heidelberg, 2012, pp. 113–120.

[11] A. Khodamoradi and et al., "S2n2: A fpga accelerator for streaming spiking neural networks," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 194–205.

[12] M. B. Milde and et al., "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system," *Frontiers in Neurorobotics*, vol. 11, 2017.

[13] E. Donati and et al., "Discrimination of EMG signals using a neuromorphic implementation of a spiking neural network," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 13, no. 5, 2019.

[14] G. Bellec and et al., "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature communications*, vol. 11, 2020.

[15] "The address-event representation communication protocol AER 0.02," Caltech internal memo, Feb. 1993. [Online]. Available: http://www.ini.uzh.ch/~amw/scx/std002.pdf

[16] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128128 1.5% contrast sensitivity 0.9% fpn 3 μs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–38, 2013.

[17] A. J.-F. et al., "A binaural neuromorphic auditory sensor for fpga: A spike signal processing approach," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 4, pp. 804–18, 2017.

[18] C. Frenkel, "Reckon: A spiking rnn processor enabling on-chip learning over second-long timescales," https://github.com/ChFrenkel/ReckOn, 2022.

[19] A. M. Devices, "Pynq-zu development board," https://xilinx.github.io/PYNQ-ZU/, 2022.

[20] Robotic and T. of Computers Lab, "Lemniscate edscorbot dataset," https://github.com/RTC-research-group/LemniscateEDScorbotDS, 2023.

[21] S. Canas-Moreno and et al., "Towards neuromorphic fpga-based infrastructures for a robotic arm." *Auton Robot*, vol. 47, pp. 947–961, 2023.

[22] wandb.ai, "Weights and biases tool," https://github.com/wandb/wandb, 2018.

Authorized licensed use limited to: TU Delft Library. Downloaded on February 26,2025 at 10:11:52 UTC from IEEE Xplore. Restrictions apply.