Review of open neuromorphic architectures and a first integration in the RISC-V PULP platform

Michelangelo Barocci*, Vittorio Fra*, Enrico Macii* and Gianvito Urgese*
* Politecnico di Torino, 10138, Torino (TO), ITALY. Email: name.surname[at]polito.it

***Abstract*—Although initially conceived as a tool to empower neuroscientific research by emulating and simulating the human brain, Spiking Neural Networks (SNNs), also known as third generation neural networks, are gaining popularity for their low-power and sparse data processing capabilities. These attributes are valuable for power-constrained edge and Internet of Things (IoT) applications. Several open-source FPGA and ASIC neuromorphic processors have been developed to explore this field, although they often require additional computing elements to manage data and communications. In this work, we review recent open-source neuromorphic architectures and the PULP ecosystem. We then present an integration of the ReckOn digital neuromorphic processor with the PULPissimo RISC-V single core microcontroller to enable edge IoT applications. Our integrated design is validated through QuestaSim hardware simulation. Through this integration of low-power neuromorphic and RISC-V processors, we focus on the promising potential of SNNs for optimizing edge IoT systems constrained by power budgets and data sparsity.**

***Index Terms*—Neuromorphic Engineering, Neuromorphic Hardware, RISC-V SoC, Configuration on Edge**

I. Introduction

Since the late 1980s, when the term *neuromorphic* was coined by Carver Mead to refer to artificial neural systems with architectures and design principles inspired by those of biological systems [1], both hardware and computation aimed at providing brain-inspired functionalities have evolved significantly [2, 3, 4, 5]. Despite this, and although the neuromorphic paradigm has demonstrated the possibility of achieving remarkable gains in terms of energy saving [6, 7, 8, 9], dedicated hardware is still hardly available. For this reason, services and infrastructures like Ebrains [10] or the Intel Neuromorphic Research Community (INRC) [11] can be joined to get access to platforms like SpiNNaker [12, 13], BrainScaleS [14] or the Loihi chip [15, 16]. Such a limitation can negatively impact the foreseen widespread of neuromorphic solutions for on-edge and low-power applications in domains like the Internet of Things (IoT) and Industry 4.0 [17, 18, 19, 20, 21, 22]. Complementary strategies are needed, especially at the research level, to foster and accelerate further development of neuro-inspired models and systems. To this aim, Field Programmable Gate Arrays (FPGAs) represent an optimal tool to prototype and deploy digital neuromorphic architectures: their flexibility makes them particularly suitable for customization and promising candidates for open-source HW [23]. However, their need for external data management and transfer forces them to consider an additional processor responsible for controlling operations during runtime [24].

FPGA-based design of neuromorphic architectures therefore implies a twofold perspective, or a co-design of collaborative units: on the one hand, the definition of the actual neuromorphic architecture and, on the other hand, the integration with a traditional processor to configure the Spiking Neural Network (SNN) and complement the data analysis in the digital world.

This paper reviews the open neuromorphic architectures suitable for open-HW design and integration with the Parallel Ultra Low Power (PULP) platform. We also show a preliminary example involving ReckOn [25] and PULPissimo [26] as neuromorphic processor and traditional micro-controller respectively.

II. Architectures overview

FPGA-based neuromorphic architectures have gained popularity due to their potential advantages for the development of dedicated HW coprocessors [27]. Recently, FPGA-deployable neuromorphic architectures have offered computer science researchers access to neuromorphic HW that may otherwise be limited. They also provide flexibility in programming the architecture to suit different use cases and experiments. Researchers can rapidly iterate on different neuron and synapse models, learning rules, and network topologies without needing to deploy a new HW design for each change. However, FPGA designs also face challenges related to resource constraints, limited efficiency, and long reconfiguration times. Nonetheless, the unique combination of accessibility, flexibility and reconfigurability offered by FPGAs has made them good candidate platforms for developing neuromorphic architectures in the newcomer of the neuromorphic research community.

Digital neuromorphic architectures that are FPGA-deployable vary widely in their design goals. For example, some architectures such as ODIN [28] and ReckOn [25] are designed as single-core solutions to increase power efficiency. On the contrary, architectures such as Minotaur [29] and Spiker [30] prioritize optimization of inference performance over energy consumption. Minotaur is a digital event-driven SNN accelerator with online learning capabilities that optimizes performance by working on memory accesses for synaptic connections, weights, and neuron statuses. Spiker, on the other hand, consists of a single layer of 400 Leaky Integrate and Fire (LIF) neurons updated with a clock-driven approach, meaning that the internal state of the neuron is updated at every clock cycle, even in the absence of spikes. Other architectures, such as SENeCA [31], are powered by a low-power RISC-V Ibex core that offers a wide range of possibilities in terms of hardware co-design, thanks to the neuron processing instruction set that it provides. In contrast, the work proposed in [32] is capable of adapting

its computing scheme between clock or event-driven based on the frequency of the input spikes. Some architectures are targeted towards scalability, like RANC [33], a scalable neuromorphic platform designed to accelerate research in the SNN domain, while others like Darwin [34], a neuromorphic co-processor powered by a RISC-V core, focus on improving computational efficiency. Finally, the architectures proposed by [35] address several critical issues pertaining to efficient parallelization of the update of membrane potentials, on-chip storage of synaptic weights, and integration of approximate arithmetic units.

From this short analysis, it is clear that there is a wide diversity of targets for these architectures, each with its unique focus. However, what truly makes a difference in this field is the availability of these architectures under open-source licensing. This availability makes these architectures appealing to use, integrate, and extend.

A. Open neuromorphic architectures

In the following, we will review in detail the four principal open-source neuromorphic architectures that can be potentially integrated with a digital microprocessor to be exploited as a neuromorphic coprocessor in IoT use cases.

1) ODIN: a digital neuromorphic processor developed by Frenkel et al. [28]. The chip is open-source and comprises N = 256 neurons and N^2 synapses arranged in a crossbar array. The ODIN processor was designed to be power-efficient and to minimize chip area overhead. The results obtained [28] demonstrate that ODIN chip consumes just $15\,\mathrm{nJ}$ for the classification inference of the MNIST dataset with 84.7% accuracy, with a power efficiency of $12.7\,\mathrm{pJ}$ per synaptic operation. The chip is fabricated using a $28\,\mathrm{nm}$ FDSOI process, occupying a total area of $0.086\,\mathrm{mm}^2$.

The chip has two on-chip SRAM modules that store the network configurations and status for neurons and synapses. Synapses are characterized by 3-bit words each, plus 1 bit to enable online learning. Neurons are described using 126-bit words, comprising 70 configuration bits, 55 status bits, and 1 bit for choosing the neuron model. ODIN has two possible event-driven neuron implementations: the Leaky Integrate-and-Fire (LIF) model and a custom model based on the 20 Izhikevich behaviors observed in biological neurons [36]. Both neuron models support online learning via the Spike Dependent Synaptic Plasticity (SDSP) learning rule, which has been proven to be convenient for chip area management while showing limited accuracy loss with respect to offline training algorithms such as Gradient Descent-based algorithms, as demonstrated in [28].

In Fig. 1 it is reported the high level architecture of the ODIN core. ODIN is equipped with a Serial Peripheral Interface (SPI) port that can be used to configure the SNN by accessing the parameters bank's write-only registers. The SPI port can also be used to access the SRAM blocks to read or write neuron statuses or synaptic weights. To execute an operation, a specific command is sent to ODIN via a 20-bit address and a 20-bit data packet, either on the MOSI channel in case of a write operation or on the MISO channel if a read operation from the SRAMs is requested.

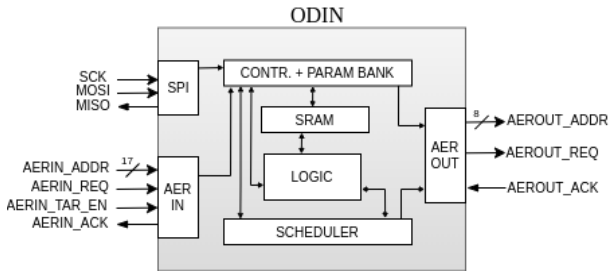

Fig. 1. Schematic representation of ODIN's architecture

The input Address Event Representation (AER) bus of ODIN has been expanded from 8 bit (256 neurons addressing) to 17 bit to increase the capabilities of the implemented SNN: a 17 bit-wide bus message can stimulate the network with five possible neural/synaptic events:

- The *Neuron spike event* is a standard synaptic operation between a pre-synaptic neuron (addressed in the bus) and all post-synaptic neurons, it triggers the update of the neurons with the associated synaptic weights.
- The *Single synapse event* is similar to the neuron spike event, but with the additional provision of the post-synaptic neuron address on the bus. Only the destination neuron is updated with the related synaptic weight.
- The *Virtual synapse event* updates the status of a synapse by providing the post-synaptic neuron address and a weight value. No source neuron is involved in this operation.
- The *Neuron time reference event* defines the time constant for the Izhikevich neuron model by providing an external time reference event to all neurons.
- The *Bistability time reference event* triggers the bistability mechanism in all synapses.

Spiking events generated by firing neurons in ODIN are managed by a scheduler that receives 14-bit long event packets. These packets contain information about the source neuron's address, the number of spikes generated in case of a burst, and the Inter-Spike-Interval.

ODIN's Address Event Representation (AER) is also utilized at the output stage. In normal operation, the controller writes the address of the source spiking neuron to the 8-bit `AEROUT_ADDR` field on the AER bus. Additionally, the same AER bus can be employed for monitoring purposes. ODIN was successfully synthesized in [24] by targeting a Xilinx PYNQ Z2 FPGA.

2) TinyODIN: a simpler and low-cost version of ODIN. Designed to reduce cost, some portions of architecture were removed, the ones related to the Izhikevich-based custom neuron model and the ones for online learning. As a matter of fact, only 12-bit LIF neuron models are available within TinyODIN.

3) ReckOn: the latest open-source neuromorphic digital processor developed by Frenkel et al. [25] that is capable of

simulating Spiking Recurrent Neural Networks. One of the key features of ReckOn is the biological plausibility provided by the online learning algorithm called e-prop [37]. This algorithm is an approximation of the BackPropagation Through Time (BPTT) training algorithm that exploits eligibility traces.

ReckOn's architecture comprises of 256 input and recurrent Leaky Integrate-and-Fire (LIF) neurons and 16 Leaky-Integrate (LI) output neurons that can be configured to perform classification or regression tasks. The LI output neurons do not have any firing and reset mechanisms.

Benchmark tests have demonstrated the low power consumption of ReckOn [25] with the best efficiency achieved at $5.3\,\mathrm{pJ}$ per synaptic operation, requiring a power budget of less than $70\,\mu\mathrm{W}$ at $0.5\,\mathrm{V}$ when processing spiking data from neuromorphic sensors. Seconds-long learning showed accuracy values of 87.3% in a 10 class classification task, 90.7% in a Keyword Spotting task, and 96.4% in a binary navigation task. As represented in Fig. 2, memory elements are arranged as follows: a parameter bank stores the network configuration parameters, three SRAMs blocks store input ($64\,\mathrm{kB}$), recurrent ($64\,\mathrm{kB}$) and output ($8\,\mathrm{kB}$) weights, and a fourth $2\,\mathrm{kB}$ SRAM stores all neurons parameters and status. Similarly to ODIN, ReckOn is provided with an SPI slave peripheral that allows configuring the network and accessing the SRAM blocks for neurons and weights data. More informations on the SPI communication can be found in section III-B.

Input spiking data are handled through a 4-phase handshake AER bus: the `AERIN_ADDR` channel shall be driven with an 8-bit address indicating the input neuron that receives the spike at a certain timestep. An additional input signal called `AERIN_TAR_EN` is used to communicate to ReckOn whether the input data represent the target neuron address or the target data for learning. The output of ReckOn follows a protocol similar to AER, indeed one 8-bit data channel and two ACK-REQ channels are used to transmit data inference results.

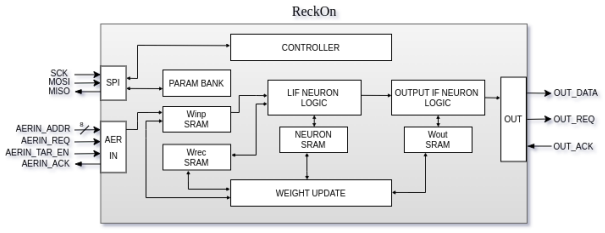

Fig. 2. Schematic representation of ReckOn's architecture

4) RANC: Reconfigurable Architecture for Neuromorphic Computing, released in 2020 by Joshua Mack et al. [33], from the University of Arizona. It consists in a neuromorphic computing framework aimed at accelerating neuromorphic computing-based research by taking inspiration from IBM's TrueNorth [38]. Respectively, three main environments are part of the RANC ecosystem: the *Training environment*, the *Simulation environment*, and the *FPGA emulation environment*. The first is used to create SNNs that can be mapped within the architecture of RANC by exploiting a backpropagation algorithm proposed in [39]. The second accurately simulates the entire RANC architecture by mapping such pre-trained SNNs. While the third environment helps to synthesize the network and to configure the memory blocks to implement the SRAM elements belonging to the neuron cores.

The architecture of RANC, whose high-level schematic can be found in Fig. 3, consists of a highly customizable 2D mesh called Network on Chip (NoC) where each basic element, a Neuron Core, is composed of five different blocks:

- The first block is the *Neuron Block*, which serves as the basic computing unit. It emulates neurons using the LIF model, and synaptic connections are mapped through a crossbar array that can be customized to have a variable number of axons (N(a)) and neurons (N(n)).
- The second block is the *Core Controller*, which is a state machine-based logic unit responsible for coordinating memory accesses and monitoring the correct functioning of the LIF neurons.
- The third block is the *Core SRAM*, which stores all configuration parameters related to each neuron inside the core. Each row in the SRAM corresponds to a specific neuron, and the dimensions of the SRAM depend on the size of the Neuron Block and the number of parameters.
- The fourth block is the *Packet Router*, which serves as the logic unit that directs output spikes' packets towards the correct destination. These packets deliver information regarding the destination core, which is computed as a N-E-S-W offset with respect to the source core. Additionally, the destination axon in that core and the time offset related to the spike event are also included in the packet.
- The fifth block is the *Packet Scheduler*, which receives the input packet from a Packet Router and sends it to the Neuron Block. This block is responsible for decoding the input spike packet into the destination axon and the time offset it should wait before feeding the spike to the axon.

RANC has demonstrated its capability to replicate IBM's TrueNorth architecture by achieving comparable accuracy results in the MNIST and EEG use cases proposed in [40], where TrueNorth was employed under the same operating conditions, network size, and encoding windows. To ensure a deterministic tick-to-tick correlation between RANC and TrueNorth, a vector matrix multiplication (VMM) algorithm was used, and RANC successfully emulated TrueNorth's behavior using the VMM mapping proposed in [41]. This benchmark has also provided insights for potential HW optimization opportunities for RANC.

Regarding the FPGA resources required to synthesize RANC, Joshua Mack et al. [33] provided a detailed analysis of the resources necessary to implement various square grids of RANC cores with 256 neurons and 256 axons. For a single-core implementation, a total of 23,152 LUTs and 5.5 RAM blocks were utilized. In contrast, for the largest square grid that can be synthesized, which is a 23x23 grid, a total of 1,599,760 LUTs and 2,684 RAM blocks were employed.

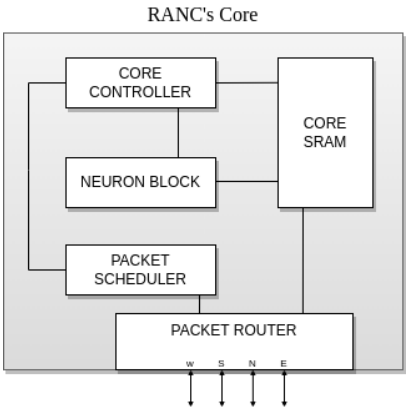

Fig. 3. Schematic representation of the Neuron Core structure of RANC

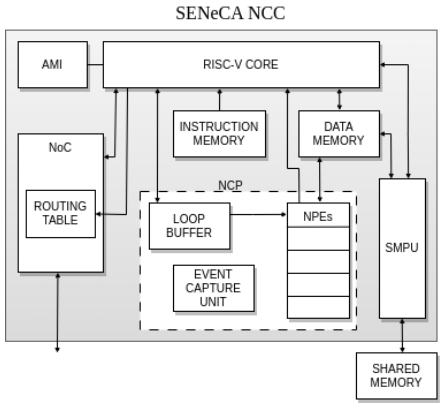

Fig. 4. Schematic representation of the NCC structure of SENeCA

5) SENeCA: Scalable Energy-efficient Neuromorphic Computing Architecture: it's an open-source digital neuromorphic processor designed to enhance the execution of SNNs on the edge [31]. One of the most significant features of SENeCA is its flexibility in terms of neuron implementation, as it incorporates a set of HW-accelerated instructions that can be used to model various types of silicon neurons and learning algorithms, instead of relying on pre-defined neuron models. This gives users the power to customize their neural network models and optimize them for their specific use cases.

SENeCA can be instantiated as a multi-core architecture, with the basic computing unit called Neuron Compute Clusters (NCC). As illustrated in Fig. 4, an NCC comprises several main elements, including:

- The *Ibex Core*: a low power, 32-bit, 2-stage pipeline RISC-V core that controls all the operations of the NCC.
- The *Axon Message Interface*, responsible for managing input and output events and filtering.
- A *Shared Memory Pre-Fetch Unit*, that is used as an advanced DMA to fetch data stored in the memory block shared between NCCs.
- A *Network-on-Chip (NoC)*, which acts as a multicast router that delivers spike events between NCCs by using stored routing tables. These routing tables allow for source-based addressing and can be remapped later by the Ibex core.
- The *Neuron Co-Processor*, which executes neuromorphic instructions inside its Neuron Processing Elements (NPE). It supports for different data types for applications that require specific quantization. Other components inside the NCP include the Event-Capture Unit, used for handling input sparse data, and a Loop Buffer that stores instructions that are going to be executed repeatedly in a register-based memory.

An implementation of SENeCA comprising one NCC with 8 NPE, 2 MB of data memory and 128 kB of instruction memory required 5k CLBs, 4 Block-RAM and 8 Ultra-RAM units on a Xilinx Virtex-7 FPGA [31].

B. Open RISC-V cores

The landscape of RISC-V based processors is seeing a relevant widening in terms of accessible architectures. In the last years many companies and research centers have been focusing on the release of RISC-V based architectures, both commercially available or open-source. This is due mainly to the open-source nature of the RISC-V ISA that allows developers to create their own architectures without worrying about licensing and by enabling collaborative communities like PULP. Besides the commercial products, our interest resides mainly in using open-source architectures, because they can be customized and tailored to each specific need. This is the case of RocketChip [42], developed by University of Berkeley, which has already been proved suitable for the kind of applications we are looking for [24], or the processors and cores developed by PULP, which are aimed at providing IoT-compatible end nodes where energy efficiency is the key requirement.

PULP is an open-source project born with the collaboration of ETH University of Zurich in Switzerland and the University of Bologna in Italy, aimed at creating low power architectures based on RISC-V Instruction Set and optimized for IoT applications [26]. Started in 2013, different open-source chips and cores have been released, with the state of the art being OPENPULP, a multi-core RISC-V architecture with L1 shared memory between cores.

PULPissimo: a 32-bit single-core, low-power microcontroller architecture released by PULP-platform in 2018 [26]. It can be implemented with either the CV32E40P (formerly RI5CY) 4-stage pipeline core or the Ibex 2-stage core, and its architecture includes advanced features such as:

- *μDMA*: an autonomous I/O subsystem that manages data transfers to and from the external peripherals [43], independently from the main core thanks to the dedicated memory allocation in the L2 block. More informations on the μDMA are listed in section III-C
- *HWPE support*: HardWare Processing Engines are application-specific components that provide dedicated HW to accelerate certain tasks. In PULP platform, HW-

PEs are tightly coupled with the on-chip memory (L2 for single-core platforms like PULPissimo or L1 in multi-core configurations), where they can access and write data efficiently.
- *Supported interfaces*: Controlled by the μDMA, there are many peripherals attached to PULPissimo: SPIM, I2C, UART and I2S and others.

III. Integrating Digital and Neuromorphic Processors

A. Neuromorphic Hardware Architectures

Table I presents a comparison of the four open-source neuromorphic architectures discussed. To achieve our goal of exploring the integration between neuromorphic and traditional processors for on-the-edge applications, it is necessary to select a specific neuromorphic architecture to use alongside the RISC-V processor. The architectures can be divided into two main groups: multi-core (RANC and SENeCA) and single-core (ODIN and ReckOn). RANC is designed as a research platform to explore various SNN architectures and is not optimized for power efficiency. In contrast, SENeCA appears to be a more comprehensive architecture, since it already includes a RISC-V core for runtime control. Since we are searching for an architecture that can coexist with a RISC-V processor that controls its configuration and operations, the single-core group of neuromorphic processors appears to be better suited for our purpose. ODIN and ReckOn employ the widely used SPI protocol for configuration and monitoring purposes, which is compatible with most digital architectures. Furthermore, the Verilog code they provide requires no modification and can be used as-is, simplifying the integration process for users.

With respect to ODIN, ReckOn has some advantages from the point of view of the architecture (refer to Table I):

- Greater number of neurons and synapses per core
- Greater energy efficiency (pJ/SOP)
- Synaptic weights are described with more bits

Even if the range of potential applications of these two architectures is different. This led to our choice to start integrating ReckOn as a neuromorphic coprocessor with the RISC-V processor.

B. ReckOn programming

In order to provide a comprehensive overview of the integration we implemented between Pulpissimo and ReckOn, it is important to first discuss the various operations needed to configure the ReckOn platform.

The ReckOn neuromorphic processor provides users with a high degree of flexibility in terms of spiking RNN configuration. Specifically, it is equipped with an external SPI slave interface that enables the master to access both the SRAM data and the parameter configuration. There are up to 60 network parameters that can be used for programming and controlling the spiking RNN inside ReckOn, identified by specific addresses. For example:

- *SPI_DO_EPROP* (ADD 09): used for enabling or disabling weights' updates powered by e-prop.
- *SPI_REGRESSION* (ADD 25): used for enabling the current task as a regression or a classification.
- *SPI_NUM_INP_NEUR* (ADD 94): used to set the number of input neurons.
- *SPI_NUM_REC_NEUR* (ADD 95): used to set the number of recurrent neurons.
- *SPI_NUM_OUT_NEUR* (ADD 96): used to set the number of output neurons.

Read and write commands from the SPI master to ReckOn should be sent through a 32-bit address, structured as follows:

- `a[31]`: R/W command - `1` for requesting a SRAM read operation, `0` for requesting a write on a SRAM or on a parameter register.
- `a[30:28]`: Target command - to identify the target for the R/W operation.
- `a[27:16]`: Consecutive R/W accesses command - to request a specific number of consecutive R/W operations, this operation is performed by automatically increasing the target address.
- `a[15:0]`: Target address - to identify a specific neuron or weight to be accessed in a SRAM or a network parameter.

Following the 32-bit address, a 32-bit data message is expected on the SPI port: either in reading mode, when ReckOn sends the SRAM accessed word through the MISO channel, or in writing when the master is expected to drive the MOSI channel for configuration.

C. PULPissimo's μDMA

The μDMA subsystem integrated in PULPissimo has been developed by the PULP-platform team to overcome traditional DMAs' data transfer bottlenecks by doubling the average bandwidth between common systems [43], this is done by tightly coupling the system with the multi-bank on-chip memory via multiple channels that are used for I/O communication and a specific handshake protocol.

Each peripheral is directly connected to the memory through the TX and RX channels, which are independent and not synchronized. In the case of SPI, there are two main TX and RX channels that have their own configuration registers that can be used to enqueue I/O transfers: starting from the base addresses of `SPIM_TX_SADDR` (`0x1A102110`) and `SPIM_RX_SADDR` (`0x1A102100`), which contain respectively the 32-bit pointer to the L2 memory allocation of the buffer where the data is or will be stored, the `SPIM_TX_SIZE` and `SPIM_RX_SIZE` registers at offset `0x4` contain the size of the buffer in bytes, for a maximum of 1 MB. The `SPIM_TX_CFG` and `SPIM_RX_CFG` registers at offset `0x8` contain other auxiliary configurations for the transfer, including the enable transfer bit.

In addition to the TX and RX channels, there is a third `CMD` channel that is used to give commands to the μDMA , at address `0x1A102120`. It has the same three configuration registers of the other channels and it's used to give specific commands to handle the transfers, like the ones needed to set the SCK frequency, transfer or receive data, repeat certain

	ODIN [28]	**ReckOn** [25]	**RANC** [33]	**SENeCA** [31]
Type of architecture	Single-core	Single-core	Multi-core	Multi-core
Supported SW framework	-	-	Platform-based	SDK
# of Neurons/core	256	256+16*	512**	256***
# of Synapses/core	64k	132k	144.9k**	-
Energy	8.4 - 12.7 pJ/SOP	5.7 pJ/SOP	-	12.7 pJ/SOP**** [44]
Neuron model	LIF + Izhikevich	LIF	LIF	Custom
Online learning	SDSP	e-prop	-	Custom
Benchmark	MNIST digits	DVS Hand Gestures Keyword spotting Navigation	MNIST digits EEG VMM*****	Human Activity Recognition Handwritten digits [44]

TABLE I
SUMMARY TABLE OF THE DATA AVAILABLE FOR THE OPEN-SOURCE NEUROMORPHIC ARCHITECTURES

*: 256 INPUT/RECURRENT NEURONS + 16 OUTPUT NEURONS.
**: 512 INPUT AXONS AND 283 OUTPUT NEURONS WAS THE CORE CONFIGURATION USED THAT MAXIMIZED A SINGLE BRAM PRIMITIVE ON THE GIVEN FPGA.
***: MAXIMUM NUMBER OF NEURONS THAT WERE USED IN A SINGLE SENECA CORE
****: ENERGY PER SOP RELATED TO AN IF NEURON.
*****: RANC'S VMM CAPABILITIES WERE BENCHMARKED USING SAR AND CIFAR-10 IMAGE RECOGNITION DATASETS.

blocks of commands, and configure the Chip Select bit when multiple slaves are connected to the peripheral.

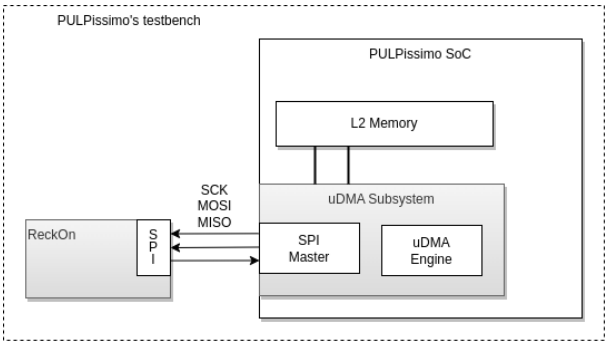

Fig. 5. Schematic representation of the ReckOn - PULPissimo integration through the SPI port connection.

D. Hardware implementation

By integrating ReckOn with PULPissimo as an external peripheral, HW-level connections are made inside the main testbench file of the processor, as depicted in Fig. 5.

pulp-runtime: To enable the programming of PULPissimo without the need for a complete SDK, a simplified runtime developed by PULP-platform is available containing the necessary functions to control all the microcontroller's functionalities via C code. This runtime includes libraries, such as `udma.h` and `udma_spim.h`, which define the required registers' addresses, offsets, and C functions to configure μDMA SPIM transfers. For example, the `plp_udma_enqueue` function can be utilized to configure μDMA transfers. This function progressively writes the `SADDR`, `SIZE`, and `CFG` registers associated with the given `RX`, `TX`, or `CMD` address.

The C code that was written takes advantage of the `pulp-runtime` libraries to correctly access and program the μDMA to send the correct SPI messages for the configuration of ReckOn.

E. Results

The validation of the correct programming of PULPissimo is made using Mentor QuestaSim 10.6 installed on Ubuntu 18.06: in Fig. 6 it's possible to identify some of ReckOn network parameters that are updated after the SPI transmission, respectively, the number of input, recurrent, and output neurons (50, 100 and 10).

`tx_data_i` is an internal signal of PULPissimo where the sequence of 32-bit packets that are sent can be identified, first the command, that includes the address of the register, than the data, i.e. the number of neurons - 1.
At the end of each 64-bit transmission, the internal signals of ReckOn `SPI_ADD` and `SPI_DATA` are updated, and at the same time the value inside the programmed register.

F. Future integration

Further improvements on the integration between neuromorphic cores and traditional processors will be made by following two possible roads: the first will be to continue on the edge applications by explore the HWPE capabilities of the PULP ecosystem, which will allow to tightly couple a neuromorphic core to the on-chip memory of the microcontrollers, the other road will involve changing the computing philosophy by exploring the computational power of neuromorphic processors, this will be possible only through the use of more performing RISC-V processors like the 64-bit CVA6, which is capable of running Linux. Both HWPE and CVA6 are described in this section.

HWPE support in PULPissimo: there is an interface available in PULPissimo for integrating co-processors as memory-coupled accelerators that are used to increase the computational capabilities of a SoC and their power efficiency when performing certain specific tasks. Data exchange is not slowed

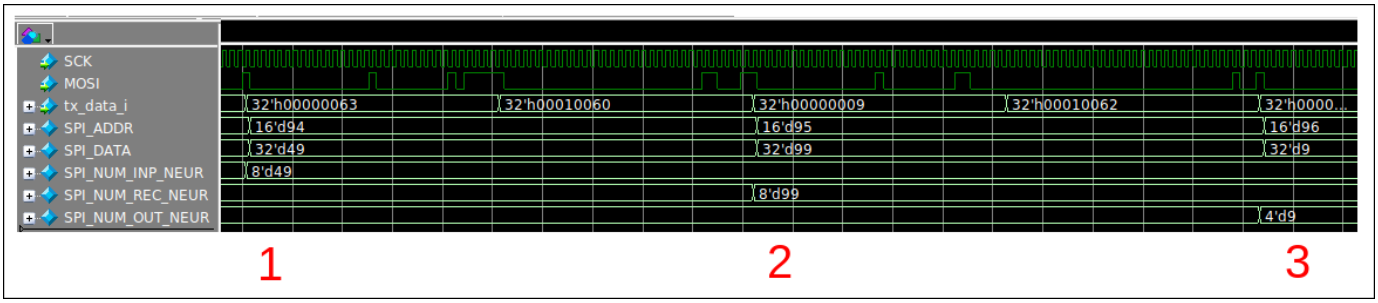

Fig. 6. SPI transmission between PULPissimo and ReckOn: as the 64-bit packets are sent from the master, the internal ReckOn registers are updated to the desired values (as specified in ReckOn's documentation, such registers should be programmed with the desired number of neurons -1):

- Timestep 1: parameter SPI_NUM_INP_NEUR at address 94 is updated with the value 49, which represents 50 input neurons.
- Timestep 2: parameter SPI_NUM_REC_NEUR at address 95 is updated with the value 99, which represents 100 recurrent neurons.
- Timestep 3: parameter SPI_NUM_OUT_NEUR at address 96 is updated with the value 9, which represents 10 input neurons.

The correct address of the parameters and the related value can be seen from SPI_ADD and SPI_DATA at each timestep, which update after every transaction.

down by additional intermediate components, indeed they are designed to work directly on the processor's memory thanks to the dedicated memory allocations, which are separated from other peripherals.
Usually the architecture of an HWPE is composed by three main sub-components and the related data exchange protocols:

- a Streamer, which handles the data transfer management to and from the L2 memory by means of the `HWPE-Mem` protocol, a 2 signals request-grant handshake to enable communication between a master and a slave. If multiple, consecutive read or write operations are needed, a more suited protocol called `HWPE-MemDecoupled` can be used to enable bursts.
- a Controller, that programs the accelerator using the `HWPE-Periph` protocol, which is very similar to the `HWPE-Mem` protocol except for additional channels used to specify the master the communication is directed to.
- the main engine, which contains the logic used to perform a specific task. There is an additional protocol called `HWPE-Stream` that allows fast movement of data inside the HWPE. This protocol is directional between a source and a sink and it uses a two signals handshake. This protocol can also be used to work on data streams, like splitting one into multiple, or merge together.

CVA6/Ariane: CVA6 is an open-source CPU based on 64-bit RISC-V ISA [45] originally developed by PULP-platform as Ariane, now maintained by Openhwgroup. The core was developed with the aim of minimizing the critical path length; this led to a 6-stage pipeline with a critical path just two times longer than server-class state of the art processors. What is peculiar about this processor is its capability to run an Operating System, like Linux, thanks to the resources overhead that were added in its development: a 39-bit page-based Virtual Memory capability, for which a Translation Lookaside Buffer (TLB) and a Page Table Walker (PTW) are implemented to accelerate the virtual to physical address translation, and large instruction and data cache elements, that led to adding critical path effects-limiting components for branch prediction, scoreboarding and other out-of-order techniques.

The authors in [45] extensively describe the power performance of the CVA6 core, by going in detail on the single 6-stage pipeline instructions. To sum up, the peak performance in terms of energy efficiency is reached at about $40\,\mathrm{GSOP\,s^{-1}\,W^{-1}}$ at $0.5\,\mathrm{V}$ with a maximum achievable frequency of less than $300\,\mathrm{MHz}$. At a given supply voltage, the maximum efficiency increases with the operating frequency.

IV. Conclusions and Future Work

In this preliminary integration between PULPissimo and ReckOn we showed how the two processors have the possibility to coexist in a single environment. This work is meant to pave the way for future implementations between the worlds of traditional and neuromorphic computing, in order to offer intelligent data elaboration in embedded applications such as IoT where power budgets are strictly limited.

For the future work we plan to expand this integration by reproducing the whole testbench proposed with the ReckOn source code, where a supervision delayed navigation problem is addressed to show the online learning capabilities of ReckOn, by performing the whole SPI configuration directly from the PULP microcontroller.

Later, once this work has been proven successful, we would like to go deeper in the integration of the two processors by exploiting the HWPE capabilities of PULPissimo. This will allow to accelerate the process of learning and performing inference by directly accessing data in the L2 memory. Also, we would like to move to other computing domains such as High Performance Computing to further develop improved hybrid neuromorphic/digital solutions.

V. Acknowledgment

This research is funded by the Ebrains-Italy project CUP B51E22000150006, the 3A-ITALY project CUP E13C22001900001 and the EU KDT JU Isolde project with Grant Agreement No. 101112274.

References

[1] Giacomo Indiveri et al. "Neuromorphic silicon neuron circuits". In: *Frontiers in neuroscience* 5 (2011), p. 73.

[2] Dmitry Ivanov et al. “Neuromorphic artificial intelligence systems”. In: *Frontiers in Neuroscience* 16 (2022), p. 1513.

[3] Danijela Marković et al. “Physics for neuromorphic computing”. In: *Nature Reviews Physics* 2.9 (2020), pp. 499–510.

[4] Jack D Kendall and Suhas Kumar. “The building blocks of a brain-inspired computer”. In: *Applied Physics Reviews* 7.1 (2020).

[5] Jia-Qin Yang et al. “Neuromorphic engineering: from biological to spike-based hardware nervous systems”. In: *Advanced Materials* 32.52 (2020), p. 2003610.

[6] Simon F Müller-Cleve et al. “Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware”. In: *Frontiers in Neuroscience* 16 (2022), p. 951164.

[7] Kyle Buettner and Alan D George. “Heartbeat classification with spiking neural networks on the loihi neuromorphic processor”. In: *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE. 2021, pp. 138–143.

[8] Emmanouil Angelidis et al. “A spiking central pattern generator for the control of a simulated lamprey robot running on SpiNNaker and Loihi neuromorphic boards”. In: *Neuromorphic Computing and Engineering* 1.1 (2021), p. 014005.

[9] Enea Ceolini et al. “Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing”. In: *Frontiers in neuroscience* 14 (2020), p. 637.

[10] URL: https://www.ebrains.eu/page/discover-ebrains.

[11] URL: https://www.intel.com/content/www/us/en/research/neuromorphic-community.html.

[12] Steve B Furber et al. “The spinnaker project”. In: *Proceedings of the IEEE* 102.5 (2014), pp. 652–665.

[13] Christian Mayr, Sebastian Hoeppner, and Steve Furber. “Spinnaker 2: A 10 million core processor system for brain simulation and machine learning”. In: *arXiv preprint arXiv:1911.02385* (2019).

[14] Sebastian Schmitt et al. “Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system”. In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 2227–2234.

[15] Mike Davies et al. “Loihi: A neuromorphic manycore processor with on-chip learning”. In: *Ieee Micro* 38.1 (2018), pp. 82–99.

[16] Garrick Orchard et al. “Efficient neuromorphic signal processing with loihi 2”. In: *2021 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE. 2021, pp. 254–259.

[17] Gianvito Urgese et al. “Powering the next-generation IoT applications: new tools and emerging technologies for the development of Neuromorphic System of Systems”. In: *Frontiers in Neuroscience* 17 (2023), p. 1197918.

[18] Kun Mean Hou et al. “Trends and Challenges in AIoT/IIoT/IoT Implementation”. In: *Sensors* 23.11 (2023), p. 5074.

[19] Vittorio Fra et al. “Human activity recognition: suitability of a neuromorphic approach for on-edge AIoT applications”. In: *Neuromorphic Computing and Engineering* 2.1 (2022), p. 014006.

[20] Evelina Forno et al. “Spike encoding techniques for IoT time-varying signals benchmarked on a neuromorphic classification task”. In: *Frontiers in Neuroscience* 16 (2022), p. 999029.

[21] Mike Davies et al. “Advancing neuromorphic computing with loihi: A survey of results and outlook”. In: *Proceedings of the IEEE* 109.5 (2021), pp. 911–934.

[22] Hongyu An, Dong Sam Ha, and Yang Yi. “Powering next-generation industry 4.0 by a self-learning and low-power neuromorphic system”. In: *Proceedings of the 7th ACM International Conference on Nanoscale Computing and Communication*. 2020, pp. 1–6.

[23] Gagan Gupta et al. “Open-source hardware: Opportunities and challenges”. In: *arXiv preprint arXiv:1606.01980* (2016).

[24] Evelina Forno et al. “Configuring an Embedded Neuromorphic co-processor using a RISC-V chip for enabling edge computing applications”. In: *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE. 2021, pp. 328–332.

[25] Charlotte Frenkel and Giacomo 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. DOI: 10.1109/ISSCC42614.2022.9731734.

[26] Pasquale Davide Schiavone et al. “Quentin: an Ultra-Low-Power PULPissimo SoC in 22nm FDX”. In: *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*. 2018, pp. 1–3. DOI: 10.1109/S3S.2018.8640145.

[27] Gianluca Bellocchi et al. “A risc-v-based fpga overlay to simplify embedded accelerator deployment”. In: *2021 24th Euromicro Conference on Digital System Design (DSD)*. IEEE. 2021, pp. 9–17.

[28] Charlotte Frenkel et al. “A 0.086-mm^2 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS”. In: *IEEE Transactions on Biomedical Circuits and Systems* 13.1 (2019), pp. 145–158. DOI: 10.1109/TBCAS.2018.2880425.

[29] Daniel Neil and Shih-Chii Liu. “Minitaur, an event-driven FPGA-based spiking network accelerator”. In: *IEEE transactions on very large scale integration (VLSI) systems* 22.12 (2014), pp. 2621–2628.

[30] Alessio Carpegna, Alessandro Savino, and Stefano Di Carlo. “Spiker: an fpga-optimized hardware accelerator for spiking neural networks”. In: *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE. 2022, pp. 14–19.

[31] Amirreza Yousefzadeh et al. “SENeCA: Scalable Energy-efficient Neuromorphic Computer Architecture”. In: *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 2022, pp. 371–374. DOI: 10.1109/AICAS54282.2022.9870025.

[32] Sixu Li et al. “A fast and energy-efficient SNN processor with adaptive clock/event-driven computation scheme and online learning”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.4 (2021), pp. 1543–1552.

[33] Joshua Mack et al. “RANC: Reconfigurable Architecture for Neuromorphic Computing”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40.11 (2021), pp. 2265–2278. DOI: 10.1109/TCAD.2020.3038151.

[34] De Ma et al. “Darwin: A neuromorphic hardware co-processor based on spiking neural networks”. In: *Journal of systems architecture* 77 (2017), pp. 43–51.

[35] Qian Wang et al. “Energy efficient parallel neuromorphic architectures with approximate arithmetic on FPGA”. In: *Neurocomputing* 221 (2017), pp. 146–158.

[36] E.M. Izhikevich. “Simple model of spiking neurons”. In: *IEEE Transactions on Neural Networks* 14.6 (2003), pp. 1569–1572. DOI: 10.1109/TNN.2003.820440.

[37] Guillaume Bellec et al. “A solution to the learning dilemma for recurrent networks of spiking neurons”. In: *Nature communications* 11.1 (2020), p. 3625.

[38] Filipp Akopyan et al. “TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.10 (2015), pp. 1537–1557. DOI: 10.1109/TCAD.2015.2474396.

[39] Steve K Esser et al. “Backpropagation for Energy-Efficient Neuromorphic Computing”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/10a5ab2db37feedfdeaab192ead4ac0e-Paper.pdf.

[40] Antonio Jimeno Yepes, Jianbin Tang, and Benjamin Scott Mashford. “Improving classification accuracy of feedforward neural networks for spiking neuromorphic chips”. In: *arXiv preprint arXiv:1705.07755* (2017).

[41] Kaitlin L Fair et al. “Sparse coding using the locally competitive algorithm on the TrueNorth neurosynaptic system”. In: *Frontiers in neuroscience* 13 (2019), p. 754.

[42] Krste Asanović et al. *The Rocket Chip Generator*. Tech. rep. UCB/EECS-2016-17. EECS Department, University of California, Berkeley, Apr. 2016. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.

[43] Antonio Pullini et al. “μDMA: An autonomous I/O subsystem for IoT end-nodes”. In: *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 2017, pp. 1–8. DOI: 10.1109/PATMOS.2017.8106971.

[44] Guangzhi Tang et al. *Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design*. 2023. arXiv: 2303.15224 [cs.NE].

[45] F. Zaruba and L. Benini. “The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.11 (Nov. 2019), pp. 2629–2640. ISSN: 1557-9999. DOI: 10.1109/TVLSI.2019.2926114.