On-Chip Learning with a 15-neuron Digital Oscillatory Neural Network Implemented on ZYNQ Processor

Madeleine Abernot, Thierry Gil, Aida Todri-Sanial

On-Chip Learning with a 15-neuron Digital Oscillatory Neural Network Implemented on ZYNQ Processor *

Madeleine Abernot
LIRMM, Univ. of Montpellier, CNRS
madeleine.abernot@lirmm.fr

Thierry Gil
LIRMM, Univ. of Montpellier, CNRS
thierry.gil@lirmm.fr

Aida Todri-Sanial
LIRMM, Univ. of Montpellier, CNRS
aida.todri@lirmm.fr

Abstract

Real-time on-chip learning is an important feature for current neuromorphic computing to enable smart embedded systems capable of learning. Neuromorphic computing based on Oscillatory Neural Networks (ONNs) are networks of coupled oscillators computing with phase information. ONNs with fully-connected connections can perform auto-associative memory applications when trained with unsupervised learning rules. In this paper, we propose for the first time an architecture to perform on-chip learning with a digitally implemented ONN. We implement the digital ONN with programmable logic of a ZYNQ processor and we perform learning on the processing system of the same chip. We validate our solution on a 15-neuron ONN trained with either Hebbian or Storkey learning rules up to three patterns. We report a stable resource utilization for both learning rules and timing from 119 μs (Hebbian) to 163 μs (Storkey). Additionally, accuracy is equal to the off-chip learning implementation.

Keywords— O scillatory Neural Networks, Auto-associative Memory, On-chip Learning, Pattern Recognition

*This work was supported by the European Union's Horizon 2020 research and innovation program, EU H2020 NEURONN project under grant n. 871501 (www.neuronn.eu)

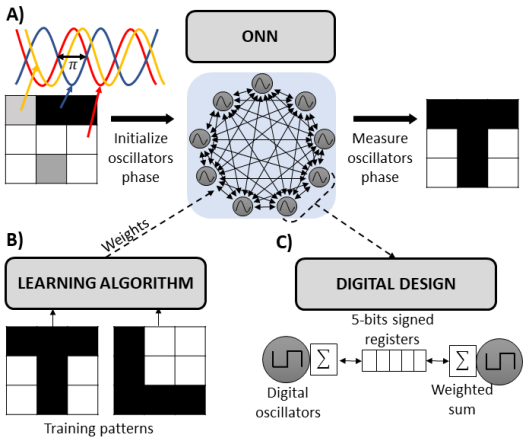

Figure 1: A) An oscillatory neural network representation with AAM type of computation, showing B) the learning task, and C) the implementation in the digital design.

1 Introduction

Neuromorphic computing aims to emulate biological neural networks and provide massive parallelism and energy efficient computing which can be a game-changer for deploying Artificial Intelligence (AI) at the edge. The human brain can easily and quickly solve complex problems by learning new information continuously. Meanwhile, deep neuromorphic models can also solve complex prob-

lems but require a tremendous amount of time, energy and resources for training. Such learning algorithms are not adapted for edge implementation. Though with the proliferation of edge devices, there is an increasing need to process data at the edge with energy efficient hardware and online learning algorithms [1].

Oscillatory Neural Networks (ONNs) are novel neuromorphic architecture enabling low-power computing suitable for edge applications [2–6]. By default, ONN is an analog computing paradigm based on coupled oscillators with phase-encoded information. Using ONNs as a fully-connected recurrent architecture, it can solve Auto-Associative Memory (AAM) problems [7]. For example, ONNs can learn patterns, such as images, and retrieve them from corrupted versions of the pattern in a few cycles, see Fig. 1. AAM tasks are typically solved using unsupervised learning algorithms, such as Hebbian or Storkey learning rules.

In this work, we propose a first on-chip learning architecture using a digital ONN design performing AAM. We use the digital ONN design from [8] and implement it in the Programmable Logic (PL) of a ZYNQ processor, see Fig. 2. Learning algorithms are implemented in the Processing System (PS), and PS controls the ONN for learning and inference. ONN receives input information or weights from PS, depending on inference or learning steps, respectively, and ONN returns its output to PS after inference computation. We implement a 5x3 ONN and two learning algorithms to test and validate our architecture. We report on resource utilization and computation time, as well as accuracy performances. In addition, we compare accuracy and resource utilization with [8] obtained on the same task with the same digital design without on-chip learning. Our contributions can be summarized as 1) design of a new architecture for on-chip learning with ONNs, 2) implementation of on-chip learning with a digital ONN using an SoC with a processing system, and 3) validation of the proposed architecture on several applications on a 5x3 ONN for pattern recognition.

Section 2 describes the ONN computing paradigm and how it performs AAM. Section 3 explains the proposed architecture to enable on-chip learning with the digital ONN design. Section 4 presents results of our proposed architecture. And Section 5 discusses the advantages and limitations of such architecture, as well as our future explorations.

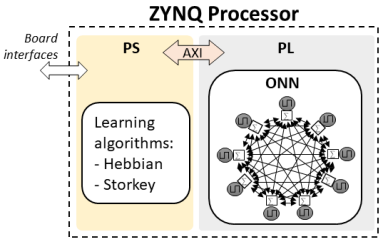

Figure 2: Global view of the on-chip learning architecture with our digital ONN design.

2 Oscillatory Neural Networks

ONNs are brain-inspired computing paradigms using the physical interaction among coupled oscillators to perform parallel computation. ONNs use oscillators as neurons and coupling as a synapse to compute with phase information, see Fig. 1.A). Information is contained in the phase difference among oscillators, so computation starts by initialization of each oscillator phase (to represent the input information). Then, coupling determines oscillators dynamics and evolution until stabilization, and measurement of the final phase states gives the output information.

ONNs with fully-connected recurrent architecture configured with specific coupling have been demonstrated to show AAM behavior [7]. AAM networks can memorize patterns and retrieve them from corrupted input information. For example, suppose one configures ONN to perform an AAM task with memorized patterns. In that case, one initializes ONN with a corrupted pattern and lets it evolve till ONN stabilizes to retrieve one of the memorized patterns, see Fig. 1.A). Memorized patterns, also called learning patterns, are stored in the synaptic weight couplings among oscillators. The learning step constitutes determining the synaptic weights and configuring the couplings, see Fig. 1.B).

State-of-the-art works perform AAM tasks with networks trained by unsupervised learning algorithms, such as Hebbian [9] or Storkey [10] learning

rules. These unsupervised learning algorithms only need learning patterns to compute the weights and configure the couplings among oscillators. In literature, so far, the reported ONN implementations do not perform on-chip learning [3]. They are usually pre-trained, and weights are computed using software algorithms and mapped inside the ONN, depending on the implementation [11]. Here, we propose for the first time an architecture that contains a digital ONN implementation and the learning algorithms to perform on-chip learning.

3 Methods

Our proposed architecture uses a ZYNQ processor consisting of PL and PS. We implement the digital ONN design using PL and the learning algorithms using PS. This section presents the digital ONN design implemented on PL, the different learning algorithms implemented on PS, the global architecture implementation, and the application we use to validate our design.

3.1 Digital ONN Design

The digital ONN design was presented in [8] as a proof of concept of the ONN as AAM for image recognition tasks. We implement this design in our architecture using the PL of a ZYNQ processor. In the design, oscillators are phase-controlled digital oscillators, oscillating between a logic '0' and a logic '1' within a 16-clock cycle period. Thus, phase input information varies from 0 corresponding to 0^o phase, up to 8 corresponding to 180^o phase, allowing for grayscale input images. Synapses are 5-bits signed registers, supporting integer weights between -15 and +15, see Fig. 1.C). In [8], weights are computed off-chip using unsupervised Hebbian and Storkey algorithms implemented on Matlab and then normalized and integrated into the digital design registers.

3.2 Learning Algorithms

To train ONN for AAM tasks, we apply unsupervised learning algorithms. Such algorithms are sorted into biologically plausible through two parameters: the locality and the incrementality. A local algorithm only needs information from neurons on both sides of the synapse to update its synaptic weight. An incremental algorithm learns patterns one by one, using the previous weight values as prior knowledge. Additionally, weight symmetry is a necessary criterion for ONN, as coupling between two neurons is bidirectional.

The most popular learning algorithm is the Hebbian learning rule [9]. However, it has limited memory capacity, so other algorithms with higher capacity were proposed, such as Storkey [10] and Pseudo-inverse [12]. Both Hebbian and Storkey are local and incremental and compute symmetric weight matrices. However, Pseudo-inverse is neither local nor incremental. Thus, we implement only Hebbian and Storkey learning rules in our architecture.

3.3 Architecture

We implement our architecture on the Zybo-Z7 development board [13]. The Zybo-Z7 development board is based on a ZYNQ processor [14]. The ZYNQ contains a dual-core Cortex-A9 processor and programmable logic equivalent to an Artix-7 FPGA. We use PL to implement the digital ONN as described in [8]. However, instead of controlling the ONN directly with PL using the Zybo-Z7 interface, we control the ONN from PS by connecting Zybo-Z7 interfaces to PS. PS also integrates the learning algorithms to update weights when a learning command is active.

In the ZYNQ processor, communication between PS and PL uses AXI4 parallel communication protocol. In our design, we use the AXI4-Lite variant to communicate between PS and PL, using PS as master and PL as a slave, see Fig. 3. AXI4-Lite transmits data words of 32-bits. PS receives input information during inference and sends it to ONN through AXI4-Lite. ONN computes and returns its output to PS. PS receives the learning command during training and uses the current input information as a new pattern to memorize. The learning algorithm computes the new weight matrix and transmits weights to PL. In PL, ONN deactivates computation to store the new weight values. After the weights update, ONN returns into inference mode until a novel training command appears in PS. Note that each AXI4-Lite communication sends 32-bits at a time. In the digital ONN design, weights are encoded with 5 bits so that we can send weights 6 by 6. With a 5x3 ONN of 15 neurons, the

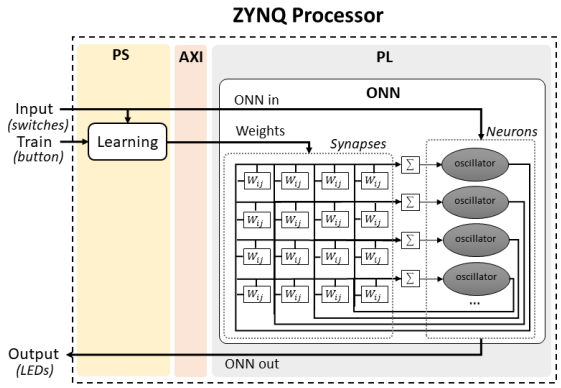

Figure 3: Architecture of the on-chip learning application with digitally implemented ONN trained by PS implemented unsupervised learning algorithms.

weight matrix contains 225 weight values. Thus, 38 AXI-Lite transmissions are necessary.

3.4 Application

We use the proposed architecture to perform real-time learning for pattern recognition with a 5x3 ONN trained with up to 3 patterns. We consider the same testbench as in [8], with three learning patterns representing digits from 0 to 2, and 15 grayscale test images. Test images are the clear learning patterns with four additional corrupted images for each pattern, see Fig. 4.

We reproduce experiments from [8], using switches to send input images and LEDs to represent the output pattern. Note, switches and LEDs are connected to PS in our work, while it was connected to PL in [8]. We initialize all weights to zeros and press a button as the learning command to start a learning process. So, when we push the training button, the PS starts processing the new weights using the current input image. After processing, all 225 weights are sent through AXI4-Lite to the ONN on the PL. During experiments, we train our model with the three training patterns one by one and we test all 15 test images to assess the accuracy.

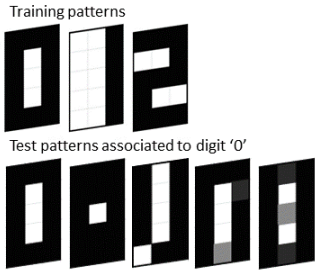

Figure 4: Training patterns and test set related to digit '0'.

4 On-Chip Learning: Results

This section reports on the performances and characteristics of our on-chip learning solution with ONN. We extract the PL resources necessary for our design, measure the time needed to learn patterns one by one, and assess the accuracy of the 5x3 digits recognition application. Additionally, we compare resources and accuracy obtained with the digital ONN design without on-chip learning [8].

Table 1 displays resources, accuracy, and timing characteristics of our on-chip learning solution for the two implemented learning rules. Table 1 also highlights resources and accuracy obtained for the same application without on-chip learning [8]. Note, we report and compare accuracy after learning the 3 memorized patterns.

Table 1 shows our solution highly increases PL resource utilization in comparison with the off-chip learning solution due to the weight variation availability. However, in our case, the learning algorithm does not influence resource utilization. Furthermore, accuracy with on-chip or off-chip learning is equal for this specific application. So, we do not lose precision with our on-chip learning architecture. Finally, we highlight the learning computation and transmission time. Computation time changes from one learning rule to another depending on the calculation complexity. However, the transmission time is stable because the number of weights sent to PL depends on ONN size (stable in this case). We report a transmission time of 86 μs to send the 225 weight values, and computation time of 33μs for the Hebbian learning rule, and 77μs for the Storkey learning rule. Note that increasing the ONN size will increase the computation and

Table 1: Performances and characteristics of the on-chip learning architecture for 5x3 digits recognition application compared to the off-chip design [8].

	Hebbian	Storkey
Resources (our work)		
- LUTs	8203 (15.42%)	8203 (15.42%)
- Flip-Flops	3305 (3.11%)	3305 (3.11%)
Resources [8]		
- LUTs	958 (1.8%)	800 (1.5%)
- Flip-Flops	721 (0.68%)	721 (0.68%)
Accuracy (our work)	93.33%	93.33%
Accuracy [8]	93.33%	93.33%
Learning time	119 μs	163 μs
- Computation	33 μs	77 μs
- Transmission	86 μs	86 μs

transmission latency.

5 Discussion

This paper presents for the first time an architecture to implement on-chip learning with a digital ONN design implemented on a ZYNQ processor. We show that the proposed architecture can efficiently perform on-chip learning for AAM tasks with a 5x3 ONN reaching equal accuracy as without on-chip learning implementation.

The proposed architecture uses a large amount of resources that can be problematic for larger scale ONN using this architecture. The amount of synaptic connections increases quadratically with the number of neurons, so a larger ONN size with on-chip learning capability can rapidly reach PL resource limitations. Yet, one advantage comes from the stability of the resource utilization when changing the learning algorithm. So, more complex learning algorithms can be implemented easily without influencing the amount of resources.

Additionally, learning needs to be fast to allow real-time weight updates depending on the application. In this paper, we demonstrate that our architecture efficiently process on-chip learning with a simple application using the Zybo-Z7 interfaces. If we consider a real-case image processing example treating images from a camera stream, real-time requirements are given by the camera. Usually 30 images per second, so 33 ms per image. In the case of small-scale ONN, such as our 15-neuron implementation, we could treat images from a camera stream in real-time. But latency will scale linearly with the number of synapses, which increases quadratically with the number of neurons. Thus, an additional study of the latency depending on ONN size should be addressed to clearly assess real-time properties of our architecture. Meanwhile, we still can improve latency of our architecture with various optimizations. For example, we can reduce the computation latency by optimizing software algorithms and decreasing the amount of sequential operations. Also, as the weight matrix is symmetric, we could send only half of the weight values to reduce by 2 the transmission latency. Finally, reducing the weight precision could allow sending more weights at a time and reduce the final transmission latency. Note that reduction of weight precision will negatively impact the accuracy, thus, there is a tradeoff between latency and accuracy with our implementation. Our future work will study the scalability of our solution with the ONN size.

6 Conclusion

We present for the first time an architecture to allow on-chip learning on a digital ONN configured for AAM tasks. In addition, we demonstrated its efficiency with a 15-neuron ONN trained with up to 3 patterns. The proposed architecture uses a digital ONN design implemented with the PL of a ZYNQ processor. In addition, we use the PS to control the ONN - send input information and request output information - and to implement two learning algorithms, Hebbian and Storkey learning rules. We use the AXI4-Lite protocol to communicate between PS and PL. We demonstrate the on-chip learning efficiency of our architecture with a 15-neuron ONN case study, trained with up to 3 patterns. We obtain equal accuracy in comparison with off-chip learning solutions. However, the new architecture necessitates a large amount of resources, limiting the implementation of larger scale ONNs. Furthermore, learning a 15-neuron pattern in this architecture takes a hundred microseconds, from 119 μs with Hebbian to 163 μs with Storkey. This respects real-time requirements for various applications, such as image recognition from a camera stream. This work proposes a solution to perform

real-time on-chip learning with small-scale ONN.

References

[1] Dennis Valbjørn Christensen and al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2022.

[2] Tetsuro Endo and Kazuhiro Takeyama. Neural network using oscillators. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 75(5):51–59, 1992.

[3] Gyorgy Csaba and Wolfgang Porod. Coupled oscillators for computing: A review and perspective. *Applied physics reviews*, 7:011302, 2020.

[4] Aida Todri-Sanial, Stefania Carapezzi, Corentin Delacour, Madeleine Abernot, Thierry Gil, Elisabetta Corti, Siegfried F. Karg, Juan Nüñez, Manuel Jiménèz, María J. Avedillo, and Bernabé Linares-Barranco. How frequency injection locking can train oscillatory neural networks to compute in phase. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2021.

[5] Juan Núñez, María J. Avedillo, Manuel Jiménez, José M. Quintana, Aida Todri-Sanial, Elisabetta Corti, Siegfried Karg, and Bernabé Linares-Barranco. Oscillatory neural networks using vo2 based phase encoded logic. *Frontiers in Neuroscience*, 15, 2021.

[6] Nikhil Shukla, Wei Yu Tsai, Matthew Jerry, Michael Barth, Vijayakrishan Narayanan, and Suman Datta. Ultra low power coupled oscillator arrays for computer vision applications. In *2016 IEEE Symposium on VLSI Technology*, pages 1–2, 2016.

[7] Frank C. Hoppensteadt and Eugene M. Izhikevich. Pattern recognition via synchronization in phase-locked loop neural networks. *IEEE Transactions on Neural Networks*, 11(3):734–738, 2000.

[8] Madeleine Abernot, Thierry Gil, Manuel Jiménez, Juan Núñez, María J. Avellido, Bernabé Linares-Barranco, Théophile Gonos, Tanguy Hardelin, and Aida Todri-Sanial. Digital implementation of oscillatory neural network for image recognition applications. *Frontiers in Neuroscience*, 15:1095, 2021.

[9] Richard G.M. Morris. D.o. hebb: The organization of behavior, wiley: New york; 1949. *Brain Research Bulletin*, 50(5):437, 1999.

[10] Amos Storkey. Increasing the capacity of a hopfield network without sacrificing functionality. In *Artificial Neural Networks — ICANN'97*, volume 1327, pages 451–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[11] Corentin Delacour and Aida Todri-Sanial. Mapping hebbian learning rules to coupling resistances for oscillatory neural networks. *Frontiers in Neuroscience*, 15:1489, 2021.

[12] Yue Wu, Jianqing Hu, Wei Wu, Yong Zhou, and K.L. Du. Storage Capacity of the Hopfield Network Associative Memory. In *2012 Fifth International Conference on Intelligent Computation Technology and Automation*, pages 330–336, Zhangjiajie, Hunan, China, January 2012. IEEE.

[13] Digilent company. Zybo z7 reference manual.

[14] AMD company Xilinx. Zynq: Socs with hardware and software programmability.