Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07220-6

ORIGINAL ARTICLE

Bio-plausible digital implementation of a reward modulated STDP synapse

Fernando M. Quintana[1] · Fernando Perez-Peña[2] · Pedro L. Galindo[1]

Received: 1 October 2021 / Accepted: 29 March 2022

Abstract
Reward-modulated Spike-Timing-Dependent Plasticity (R-STDP) is a learning method for Spiking Neural Network (SNN) that makes use of an external learning signal to modulate the synaptic plasticity produced by Spike-Timing-Dependent Plasticity (STDP). Combining the advantages of reinforcement learning and the biological plausibility of STDP, online learning on SNN in real-world scenarios can be applied. This paper presents a fully digital architecture, implemented on an Field-Programmable Gate Array (FPGA), including the R-STDP learning mechanism in a SNN. The hardware results obtained are comparable to the software simulations results using the Brian2 simulator. The maximum error is of 0.083 when a 14-bits fix-point precision is used in realtime. The presented architecture shows an accuracy of 95% when tested in an obstacle avoidance problem on mobile robotics with a minimum use of resources.

Keywords R-STDP · STDP · Synaptic plasticity · Neuromorphic system · FPGA · Spiking neural network

1 Introduction

Digital neuromorphic devices are designed to emulate the biological features and information processing capabilities of the neural structures present in the brain, by simulating SNNs [6, 11, 15]. In these neural networks, populations of neurons are connected by synapses that could be potentiated or depressed according to several conditions. This process of changing the strength of the connection (also known as synaptic plasticity) is the learning process.

Since the output of a neuron is a temporal sequence of spikes, timing plays a major role in the output features. There is evidence of synaptic adaptation between pre- and post-synaptic neurons that depends on the difference in firing time between them, following Hebbian learning behaviour called STDP [7]. STDP could have different forms of synaptic plasticity or such as classical STDP, bidirectional STDP, classical STDP with additional (LTD) [4]. One of the most commonly used forms in SNN is the pair-based model of STDP, where the strength of the synaptic connection is potentiated when a pre-synaptic spike precedes a post-synaptic one, whereas if a post-synaptic spike precedes a pre-synaptic one, the synapse is depressed [7].

STDP is an unsupervised learning method as it changes the synapse weights based on the specific spike times of pre- and post- synaptic neurons. In some cases, this learning technique is not enough and it is necessary to apply reinforcement learning strategies too to improve the performance of the application. Examples of this are in the field of robotics: a line follower vehicle [1], snake-like robotics [3], obstacle avoidance [2] or target reaching motion with a robotic arm [18]. The authors of [13] proposed a method of R-STDP based on the combination of a neuromodulator (dopamine) and the signal produced by the

✉ Fernando M. Quintana
fernando.quintana@uca.es

Fernando Perez-Peña
fernandoperez.pena@uca.es

Pedro L. Galindo
pedro.galindo@uca.es

[1] Department of Computer Science and Engineering, Universidad de Cádiz, Escuela Superior de Ingeniería, Avda. Universidad de Cádiz, n° 10, Puerto Real 11519, Cádiz, Spain

[2] Department of Automation, Electronics and Computer Architecture and Networks, Universidad de Cádiz, Escuela Superior de Ingeniería, Avda. Universidad de Cádiz, n° 10, Puerto Real 11519, Cádiz, Spain

Published online: 26 April 2022

STDP. In biology, neurotransmitters such as dopamine act as a reward prediction error signal, providing a mechanism for global modification of synapses.

This paper presents the design and implementation of a digital circuit modelling an R-STDP mechanism for synaptic plasticity. The architecture is deployed into a digital portable platform such as an FPGA. It may be used in real-time applications, being therefore of great interest in many fields, such as in artificial vision, automatic control or robotics.

Several authors have proposed different implementations for learning synapses in digital and analog hardware. As an example, Yousefzadeh, et al. used a 1-bit weight using stochastic STDP, where the weights only have two states"1"(ON) or"0"(OFF), based on the probability of the STDP function. Given a positive probability $0 \leq p \leq 1$ (Long-Term Potentiation (LTP)), after generating a random number $0 \leq x \leq 1$, if $x \leq p$ the weight will be set to"1", otherwise it will be unchanged [19]. In the case of LTD, the STDP function will generate a negative probability, so if a random number $x \leq |p|$, the weight will be set to"0", otherwise it will be unchanged. On the other hand, the authors of [5] created a minimum complexity implementation of STDP using combinational blocks, where each one represents an increase or decrease in the weight, based on the relative spike timing of the pre and post synaptic spikes. In [9], the authors created a quantized/binarized version of STDP for online learning in SNN, saving hardware resources. These methods, although efficient, do not take into account the external influence of a neuromodulator such as the dopamine, like the one proposed by Izhikevich [13]. By using an external neuromodulator, such as dopamine, certain fIzhikevichiring patterns can be reinforced even when the reward signal is delayed in time by seconds.

The system presented in this paper, ports a biological plausible approach of learning in SNN to a neuromorphic architecture based on an external learning signal that emulate the concentration of dopamine in the network. This system is implemented on an FPGA.

Subsequently, the paper is structured as follows. In Sect. 2 the neuron model to be used is defined, as well as the process of synaptic adaptation. In this section, the implementation followed to adapt, keeping the main biological features, the STDP synaptic plasticity to an FPGA. will be explained. In Sect. 3 the results obtained with our implementation on the FPGA. are presented and compared with software simulations to evaluate the accuracy of the implementation. Finally, the conclusions are presented.

2 Materials and methods

2.1 Neuron model

The Izhikevich model is used in order to replicate as many biological output features as possible [12]. This model is defined by Eqs. 1, 2 and 3

$$\dot{v} = 0.04v^2 + 5v + 140 - u + I \tag{1}$$

$$\dot{u} = a(bv - u) \tag{2}$$

$$\text{if } v \geq 30 \text{ mV} \quad \text{then } v = c, u = d + u \tag{3}$$

A fixed-point representation was used where the values of the neurons can be represented in the range [-1,1]. This is done since signed floating-point arithmetic is computationally expensive in an FPGA. In order to reduce the number of multipliers needed for the equation, the method proposed in [10] was followed. A timestep power of 2 like $dt = 1/8$ was proposed, so the operation of integration is reduced to a right shift. As the voltage is in the range [−1,1], the first constant of the equation (0.04) is scaled up by 100, because squaring the v scaled, drops the value by 10,000. This modification also scaled up the constant 0.04 to 4, which gives the chance of changing the multiplier of that constant with another right shift. Thus, the equations can be transformed as follows:

$$v = (4v^2 + 4v + v + 1.4 - u + I)/8 \tag{4}$$

$$u = a(bv - u)/8 \tag{5}$$

$$\text{if } v \geq 0.30 \text{ mV} \quad \text{then } v = c, \quad u = d + u \tag{6}$$

2.2 Synaptic model

The input current to the neuron is modeled as an alpha function that can be simplified to an exponential function, as expressed in the Eq. 7.

$$I(t) = (t/\tau_I)e^{-t/tau_I} \tag{7}$$

I represents the current that is injected into the post-synaptic neuron and τ_I is the time constant for the alpha shaped function.

The synaptic weights will be updated using STDP, which is a proven physiological process that adapts the synaptic strength depending on the firing times of the pre- and post-synaptic neurons. It is defined as a function of the sum over all pre- and post-synaptic spike times. Equations 8–10 show this behavior [1].

$$\Delta t = t_{post} - t_{pre} \tag{8}$$

$$W(\Delta t) = \begin{cases} A_{pre} e^{-\Delta t/\tau_{pre}} & if \quad \Delta t >= 0 \\ -A_{post} e^{\Delta t/\tau_{post}} & if \quad \Delta t < 0 \end{cases} \tag{9}$$

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(\Delta t) \tag{10}$$

In these equations, Δt is defined as the time difference between the pre- and post-synaptic spikes. This difference is then used in Eq. 9, where A_{pre} together with A_{post} (constant values) define the increase or decrease of the synaptic weight. Therefore, Δw is the change of the synaptic weight.

To implement this mechanism in hardware, a trace-based approach can be used [16]. This mechanism includes two variables (A_{pre} and A_{post}) that act as traces recording the spiking activity of the presynaptic and postsynaptic neuron respectively. Equations 11, 12 show this behaviour.

$$\dot{Weight}_+ = -Weight_+/\tau_+ + A_{pre} \tag{11}$$

$$\dot{Weight}_- = -Weight_-/\tau_- + A_{post} \tag{12}$$

Each time a pre-synaptic spike is produced, the weight is updated by the value of $Weight_-$ and each time the post-synaptic spike occurs, the weight will be increased by the value of $Weight_+$, as shown in Fig. 1

2.3 Reinforcement learning

As pair-based STDP relies only on the relationship between pre- and post-synaptic firing time to update the weights, it is an unsupervised learning technique.

However, in a reinforcement learning system, an agent performs certain actions in an environment that may result in a reward. The overall objective is to learn how to achieve the maximum amount of rewards. In biology, this analogy can be made with dopamine, where its functions include the reward (an external learning signal) when something is achieved. Thus, in a R-STDP by dopamine signaling, an eligibility trace keeps track of the STDP events, representing the activation of an enzyme important for plasticity [13].

Equations 13 and 14 [13] describe the behaviour of the eligibility trace and the synaptic dynamics. c is the eligibility trace, w is the synapse strength, τ_c is the decay constant, and $s_{pre/post}$ is the spike time of the pre- or post-synaptic neuron. The eligibility trace c decays with an exponential time in the range of $\tau_c = 1s$, as defined in [13]. The change of the synaptic weight w is produced by the eligibility trace c (specific to each synapse) gated by the extracellular concentration of dopamine d. Each time a reward signal is produced, the global concentration of dopamine in the network, shared by all the synapses, is increased and it will decay with an exponential time constant τ_d towards a baseline value defined by $DA(t)$. Equation 15 shows this mechanism [13].

$$\dot{c} = -\frac{c}{\tau_c} + W(\Delta t)\delta(t - s_{pre/post}) \tag{13}$$

$$\dot{w} = cd \tag{14}$$

$$\dot{d} = -d/\tau_d + DA(t) \tag{15}$$

The learning process can be stopped if the concentration of dopamine in the network is zero, as the synapse plasticity depends on the spiking activity of the neurons and the concentration of the dopamine in the neural structure.

2.4 Digital implementation

In order to implement the learning approach described in a hardware platform, some optimization techniques have been used, such as fixed point arithmetic, the use of a (LUT) to store complex calculations and the use of arithmetic right shift to divide by a power of 2, as mentioned in neuron model 2.1.

2.4.1 R-STDP full architecture

Each neuron can have a configurable number of synaptic connections as input that are integrated in a current signal with a defined shape (alpha, exp, bi-exp). These synapses are modified according to Eqs. 13, 14 and 15 that creates an eligibility trace which will be used to modify the synaptic weight based on the global dopamine concentration in the network. In each simulation timestep, the system calculates the new weight value for all the post-synaptic neuron inputs, in a time-multiplexing manner.

2.4.2 Eligibility trace

The eligibility trace (described by Eq. 13) is local to each synapse and depends on the values of the STDP. In each timestep, the STDP value from two traces that store the time difference between the pre- and post-synaptic spikes is computed. The STDP computation is made considering each input synapse population independently, taking advantage of the parallel processing capabilities of the FPGA. As the STDP is calculated per synapse, a time-multiplexing approach is used for each input synapse of the post-synaptic neuron, reducing the number of components necessary for the STDP. Each STDP component has a *weight_plus*, a *weight_minus* and a *e_trace* 14 bits register. When the synapse receives an input spike, the *weight_plus* register is increment by a fixed amount defined as a constant by the user. The *e_trace* register accumulates the value of *weight_minus*. In case a post-synaptic spike is produced, the *weigth_minus* register is increased and

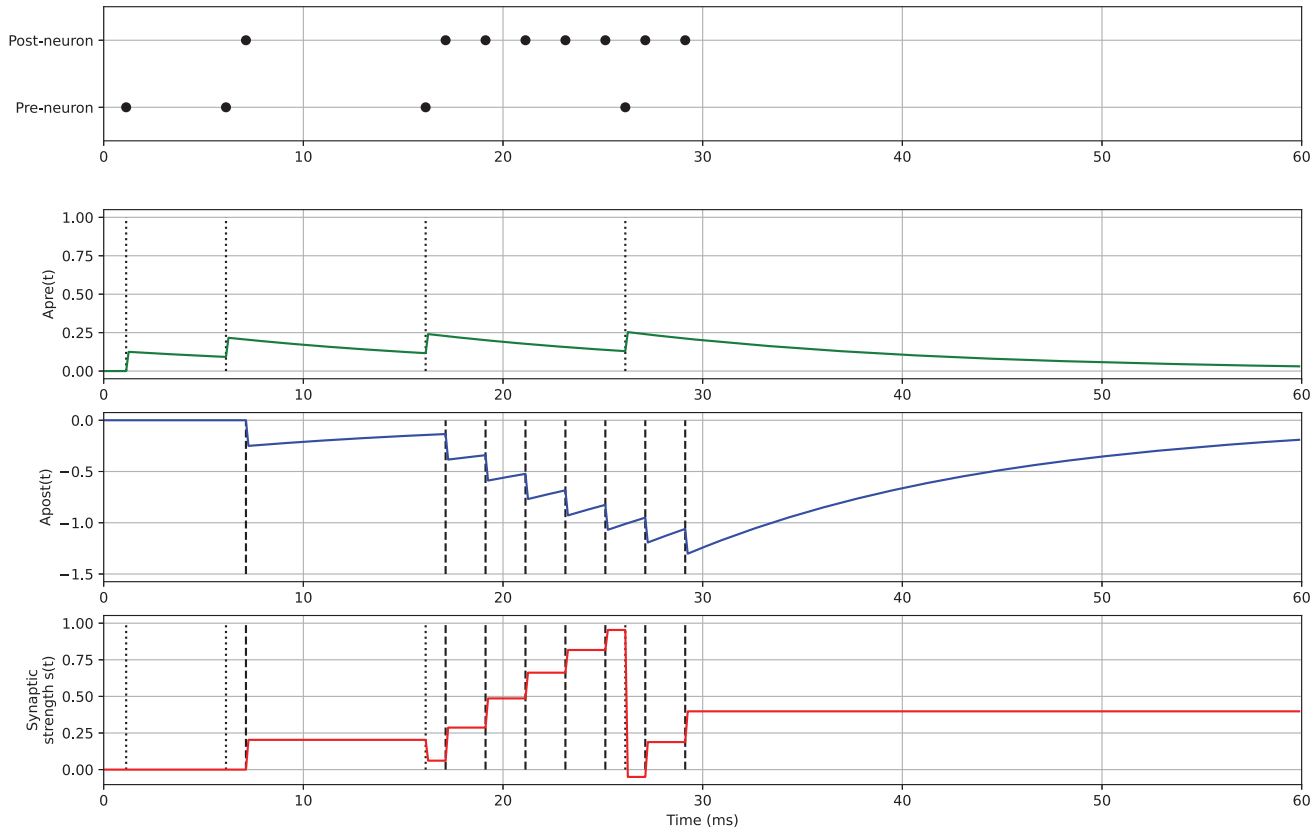

Fig. 1 STDP traces showing the temporal evolution of different signals across 60 ms of simulation time: (i) Pre and post-neuron activation, (ii) Increase of the synaptic weight (iii) decrease of the synaptic weight and (iv) synaptic strength

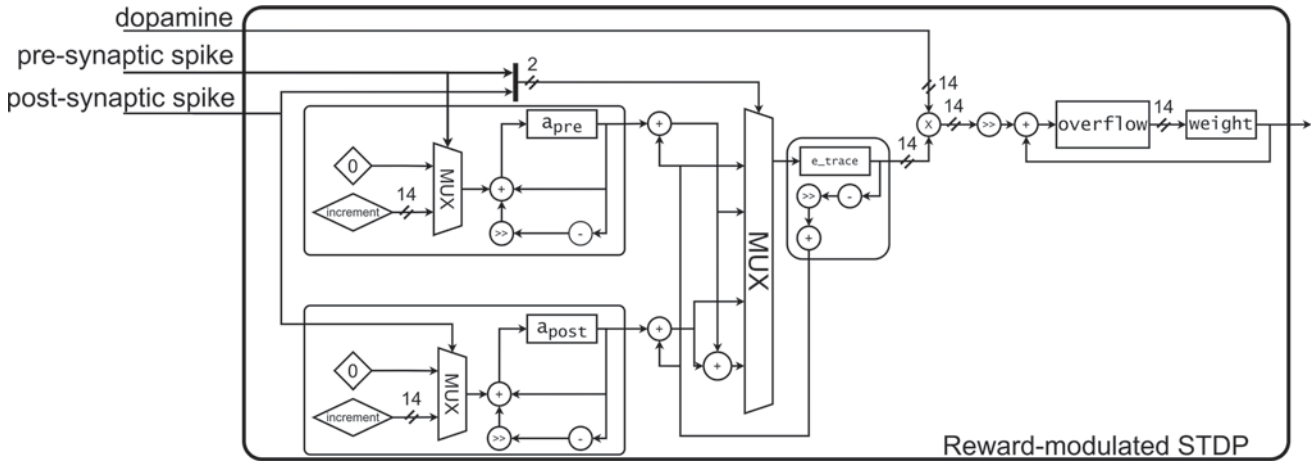

Fig. 2 R-STDP trace calculation component

e_trace accumulates the value of the *weigth_plus* register. In each timestep, all of these registers are updated using the Euler method of the differential equation defined in Eq. 13.

This process is shown in Fig. 3a, b. The yellow block is the multiplexer that decides if the trace is increased when an input spike is received. The blue block includes the dynamics of the *weigth_minus* and *weigth_plus* variables. In Fig. 3a, the blue block represents the dynamics of the eligibility trace that follows Eq. 13 and the yellow block is used to select the input of the trace.

After the e_trace is computed, the synaptic modification is computed by multiplying the trace by the global

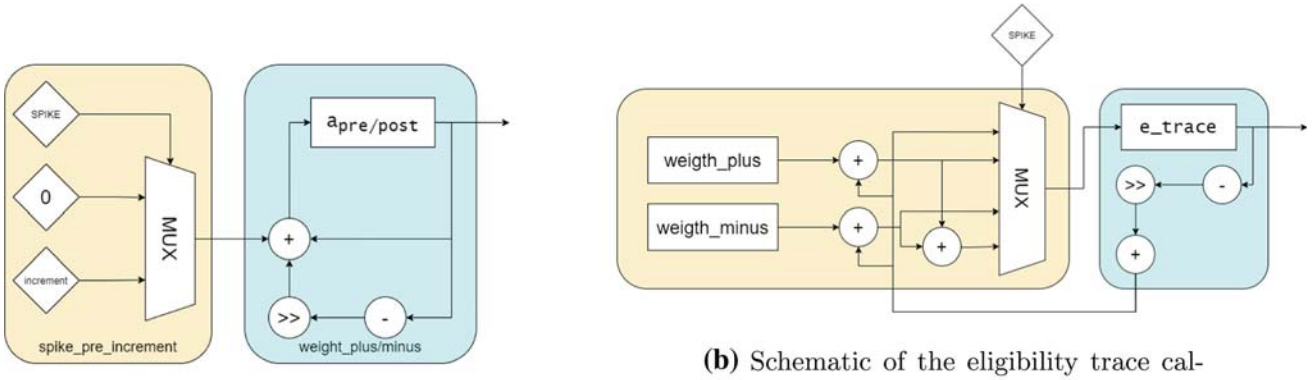

Fig. 3 FPGA modules for the implementation of the STDP eligility traces. **a** The pre- and post-synaptic traces are calculated. **b** The traces obtained in (**a**) are used to calculate the eligibility trace

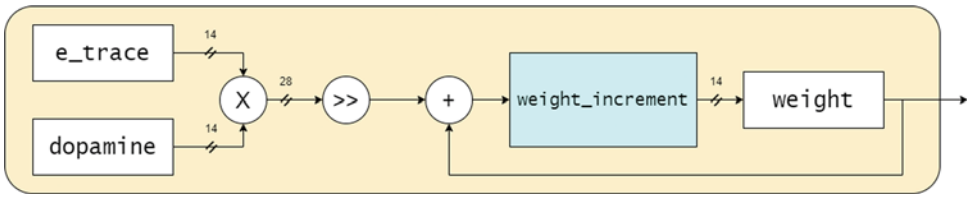

Fig. 4 Schematic of the synaptic weight modification based on the global dopamine concentration and the local eligibility trace

dopamine concentration. When two elements are added or multiplied, the possible overflow caused by the fixed-point representation is taken into account. In order to solve this problem, a module that verify if an overflow is produced is implemented (Fig. 4). In the case that an overflow/underflow is produced, the module will output the maximum/minimum value possible defined as a constant, i.e., there is a saturation mechanism implemented.

2.4.3 Synapse module

The synapse module linearly accumulates all the inputs of the post-synaptic neuron into a current signal that can have any shape defined by the user (alpha, exponential, biexponential, etc.). The base current signal shape has been precalculated and stored in memory. This signal is modified (scaled) according to the synapse weight value and modified using the R-STDP rule. Each individual connection stores the values of the eligibility trace and the pre-synaptic trace. Regarding the post-synaptic spikes of the neuron, there is only one register that stores the post-synaptic trace for all the input synapses. The synaptic module uses time-multiplexing to calculate the new value of the weight based on the R-STDP (Eqs. 11, 12, 13 and 14) within one clock cycle. The value of the dopamine concentration in the neural network, the post-synaptic spike and all the pre-synaptic spikes, are received as inputs for this module.

3 Results

The proposed architecture has been implemented on a Nexys 4 DDR board, which includes an Artix-7 FPGA.- Three experiments have been performed: (1) to check that the behaviour of the system matches the software simulations, (2) to check that the saturation mechanisms work properly,and (3) to check that the system achieves a better performance than previous experimental works on a real robotic platform: an obstacle avoidance problem within mobile robots. Furthermore, the number of resources of the FPGA needed both, for the R-STDP component isolated and for an entire neural network are shown in Tables 3 and 4.

3.1 Experiments

In the first experiment, one neuron with an input synapse that receives specific spiking impulses across time is implemented. The behaviour of the system is then compared with software simulations[1] using the Brian2 simulator [17] and the parameters described in Table 1. Figure 5 shows the difference between the software simulation and the hardware implementation (dashed line). Table 2 shows the maximum error over 60ms between both, using a 14 and 18 bits representation for the synaptic values. Figures 6 and 7 shows the error over time between the software simulation and the FPGA. These values are the result of the

[1] Github with the software simulations code.

Table 1 Simulation parameters

Parameter	Value
Pre-synaptic trace increase (tau_{pre})	16 ms
Post-synaptic trace time constant (tau_{post})	16 ms
Pre-synaptic trace increase (A_{pre})	0.125
Post-synaptic trace increase (A_{post})	− 0.25
Eligibility trace time constant (tau_c)	256 ms
Dopamine signal time constant (tau_d)	1 ms
Synapse time constant (tau_s)	1 ms
Dopamine concentration (DA)	1

Table 2 Maximum error between the simulation in Brian2 and the FPGA implementation

	Error	
Signal	14 bits	18 bits
Dopamine	9.648e-04	6.677e-05
Elegibility trace	0.083	0.011
$Weight_+$	0.017	0.001
$Weight_-$	0.015	0.001
Weight	0.019	0.005

precision of the fixed-point representation of the numerical operations. The eligibility trace has the largest error, but this is because it depends on the $weight_+$ and $weight_-$ values, thus accumulating its errors as well. However, the maximum error for 14 bits obtained in 60 ms is 0.083 being on a scale of values in the range [− 1, 1].

In the second experiment shown in Fig. 8, the correct operation of the saturation module for the weight value is checked. For this purpose, an over/underflow has been

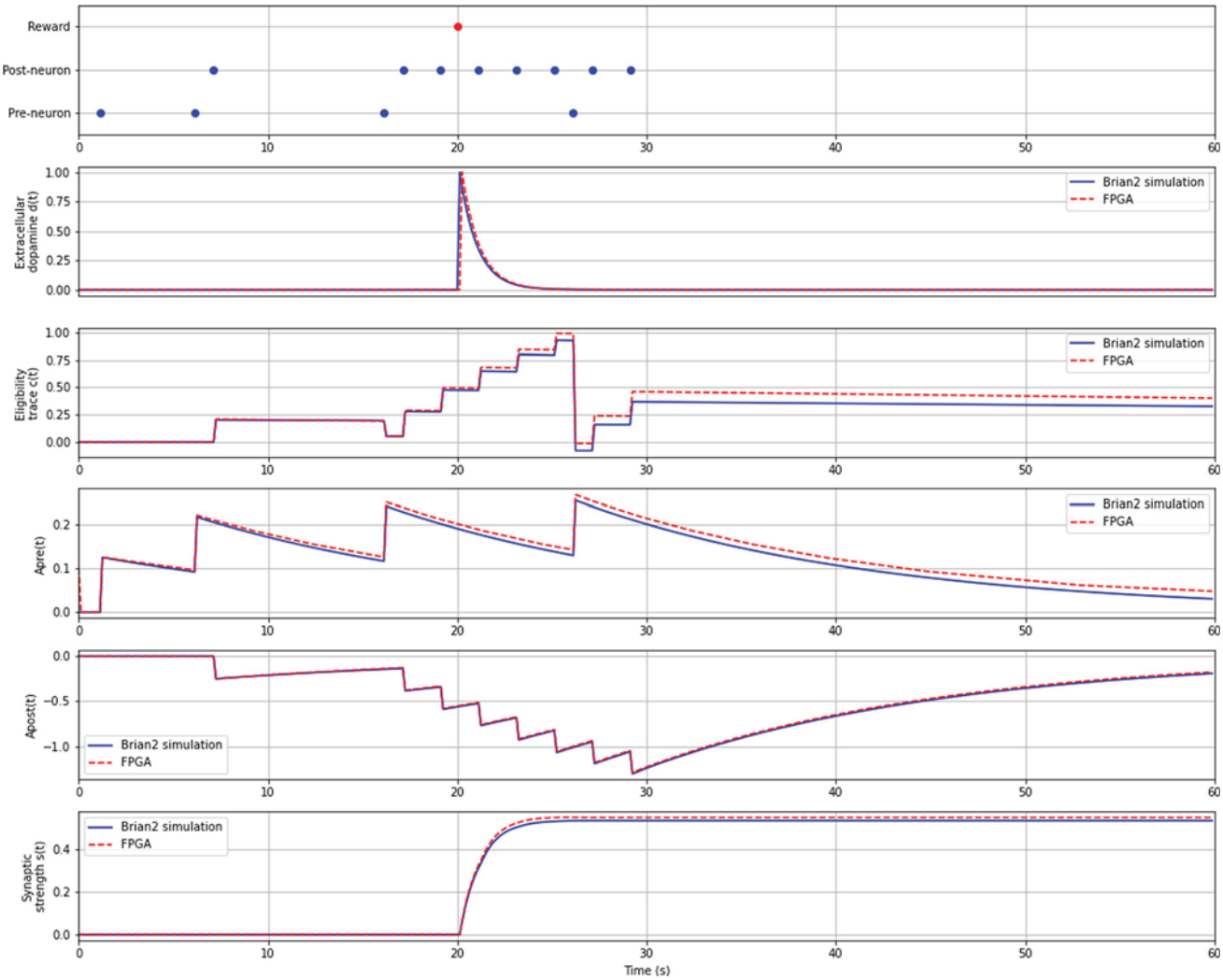

Fig. 5 Comparison between Brian2 simulation of the extracellular dopamine concentration, the Eligibility trace, Apre and Apost signal of the STDP and the weight change using R-STDP. The FPGA values are represented as a dashed line

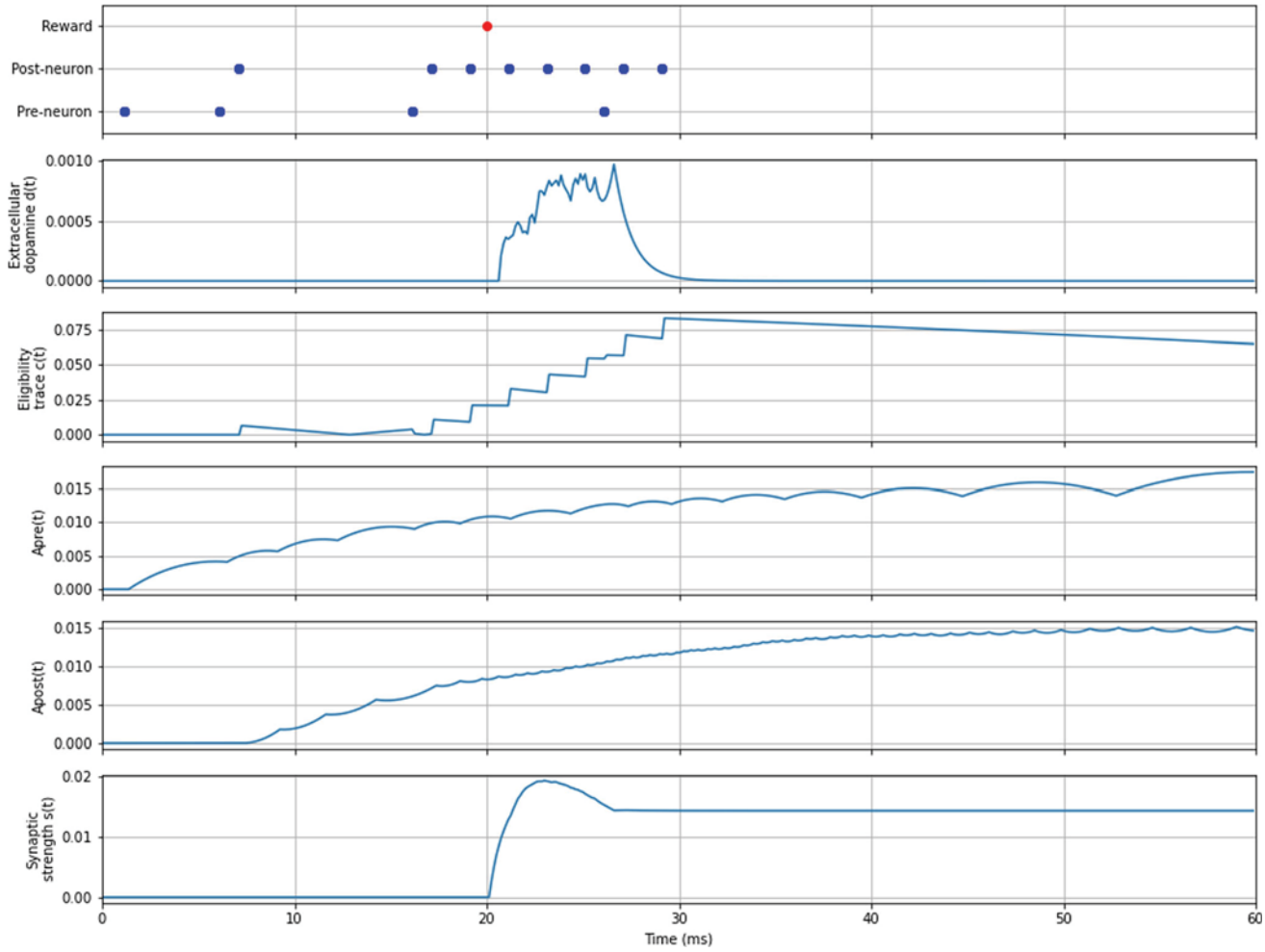

Fig. 6 The error produced over time between the Brian2 simulation and the data obtained from the FPGA using 14 bits fixed-point numerical representation

Table 3 Utilization of the synapse component using the FPGA

Resource	Utilization	Available	Utilization %
LUT	394	63,400	0.621
FF	133	126,800	0.105
BRAM	0.5	135	0.370
DSP	2	240	0.833
IO	60	210	28.571

Table 4 Utilization of the R-STDP using the FPGA

Resource	Utilization	Available	Utilization %
LUT	183	63,400	0.2887
FF	80	126,800	0.063
BRAM	0	135	0.0
DSP	1	240	0.4167
IO	41	210	19.524

provoked by forcing the number of pre/post-synaptic spikes so that the eligibility trace is negative/positive and underflows/overflow the weight value, set in a range of $[-1, 1]$.

The implemented architecture defines each neuron as an independent component physically interconnected within the FPGA. If the neural network needed on the FPGA is a large one with all the neurons in an all-to-all connection, it may happen that the timing requirements of the FPGA are not met. This is because in a all-to-all connected network, the delay between two electronic components in the FPGA could be greater than the duty cycle. For that reason, the clock frequency of the FPGA was reduced to 50MHz. Thus, the timing requirements can be met when a large number of all-to-all connected components can be achieved.

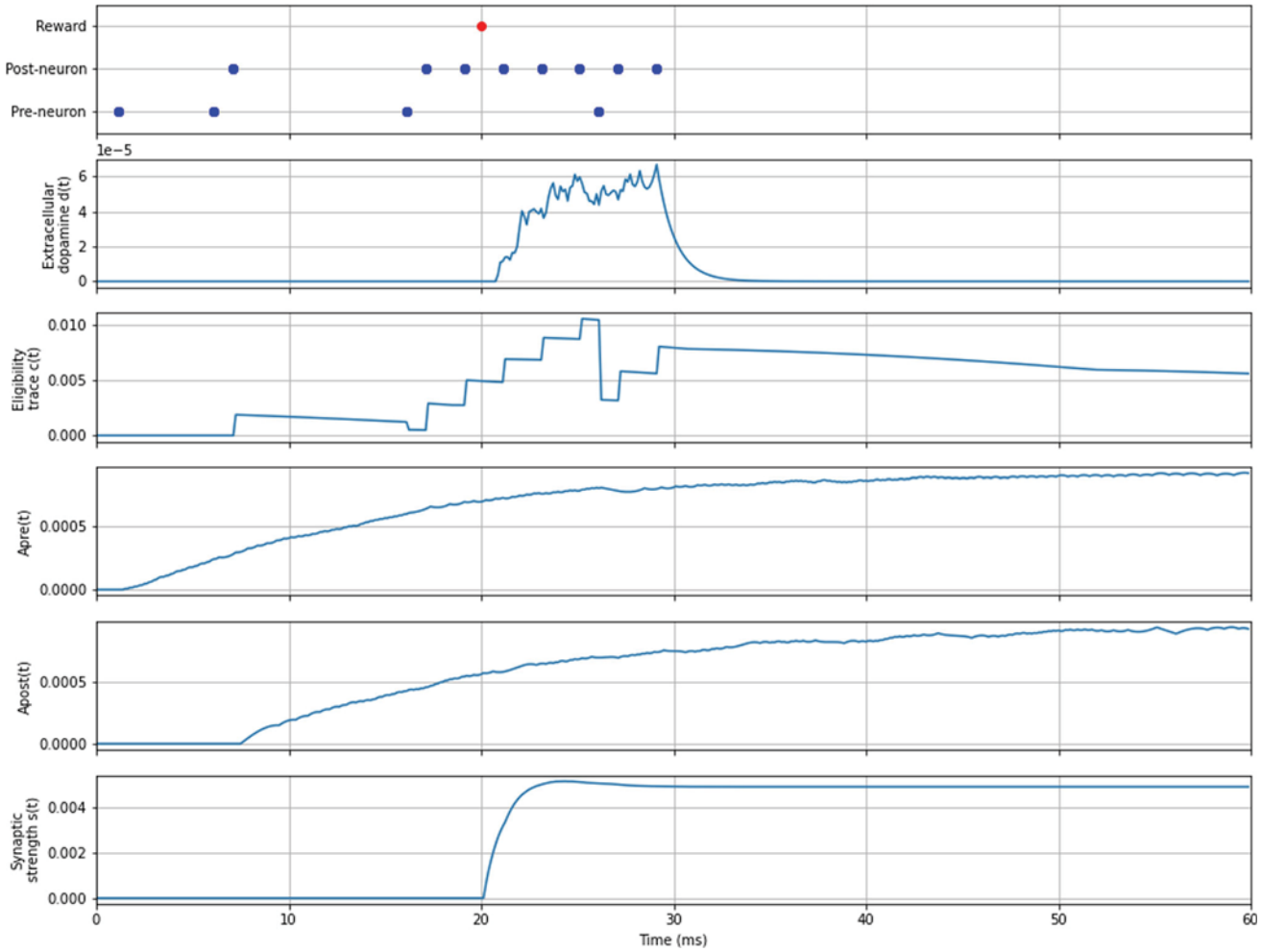

Fig. 7 The error produced over time between the Brian2 simulation and the data obtained from the FPGA using 18 bits fixed-point numerical representation

In the third experiment, an entire spiking network architecture is deployed to be used within an avoidance obstacle problem in a mobile robot. This experiment is performed to show that the system proposed can be used within the field of robotics. The experiment is based on [2] which simulate a fully connected network with a hidden layer of 50 neuron to calculate the angle at which the vehicle should turn. In our case, some modifications has been done: (1) the output neurons are connected directly to the input sensors neurons. Thus, there is no need to use a hidden layer. (2) the output is normalized and represent the speed that the motor should have in order to turn in a certain direction, instead of the approach of the original paper where the output of the neurons represented the angle (positive or negative) that the robot should turn.

The experiment consists of a mobile robot with six ultrasonic sensors attached to its external bumper and a SNN with six input Integrate and Fire (I&F) neurons and two outputs neurons (one per motor) (Fig. 9). The synapses of the network follows a biexponential current base synapse with a decay time of 3ms. Each 64ms the normalized turning direction and value are calculated based on the spike times of both output neurons (instead of the exact angle as in [2]). The output values are then compared to the one that should be obtained using the braitenberg algorithm. The dopamine concentration value is the result of such difference.

For running the SNN a timestep of 1ms has been used, as originally in [2]. Also the SNN uses a 18-bits fixed-point precision, using 10 bits for the fractional part. In order to train the network, prerecorded values for the sensors and the output that the neuron should produce has been created using the braitenberg algorithm. These values are used as input for the network in each 64ms of simulation. At the end of 6 seconds of simulation, the system achieved a 95% of accuracy (choosing the correct turning direction). The final weight distribution and the accuracy can be seen on Figs. 10 and 11. It can be seen that for the left side output

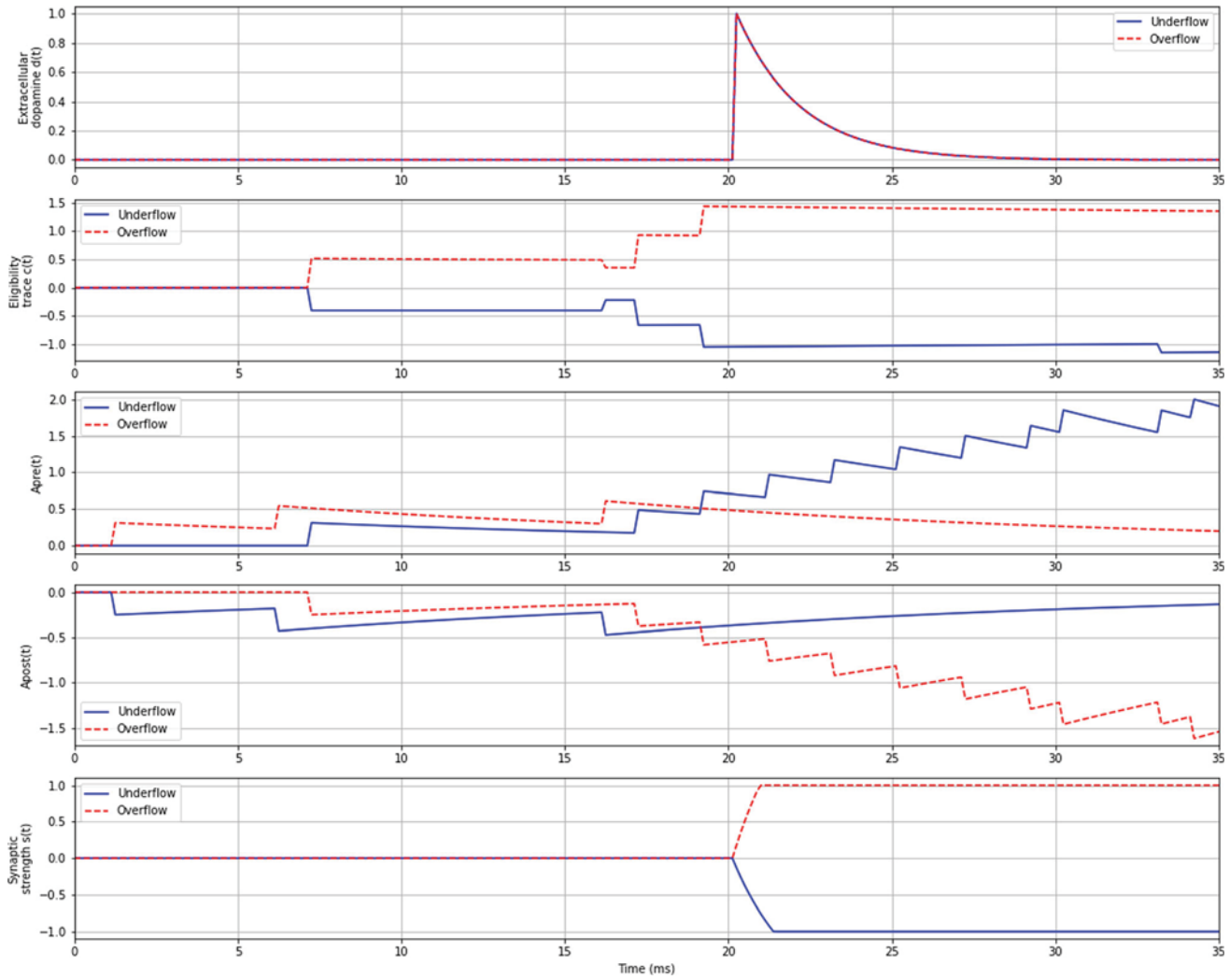

Fig. 8 Saturation experiment for the 14 bits weight value. The plots show the dopamine concentration, eligibility trace, pre- and post-synaptic traces and the weight value

neuron, the active synapses are those that correspond to the left side sensors, and vice versa for the right side motor neuron.

3.2 Comparison

A comparison between different learning rule implementations on FPGA is made in Table 5. A briefly summary of the papers used for the comparison follows: (1) In [8], the authors present the implementation of the Izhikevich neuron model and the STDP learning rule using the CORDIC algorithm. This implementation offers a better performance and higher accuracy than the original STDP model; (2) In [5], the authors present a minimum complexity implementation of STDP using combinational blocks that represents the weights changes with respect to relative spike timing of pre and post synaptic spikes; (3) In [19], the authors present a 1-bit weight implementation using stochastic STDP. This weight is updated using the probability of the STDP function; (4) In [9], the authors present an implementation of weight quantization/binarization with STDP. This implementation aims to reduce storage requirements and increase computing efficiency without a significant loss in terms of accuracy, with respect to a non-quantized version of STDP; (5) In [14], the authors present an efficient implementation of pair-based STDP (PSTDP) and triplet-based STDP (TSTDP). In the case of the TSTDP, the learning rule is based on additional traces of the pre and post synaptic spikes, taking into account interactions among triplets of spikes. For the digital implementation of both rules, a 16-bits fix-point representation and techniques of bit-shifting and Piece-wise Linear (PWL) approximations are used. The main advantage of the system implemented with respect to other methods are: (1) the addition of an external learning signal, that represents a neuromodulator (dopamine) that could

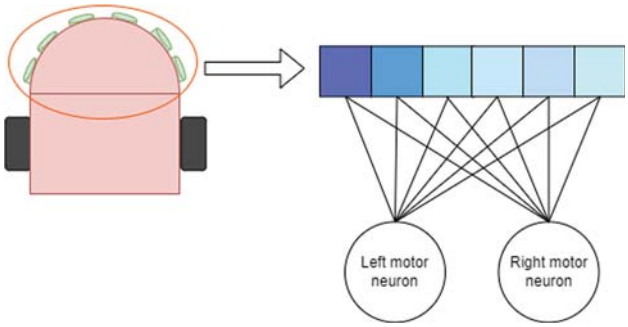

Fig. 9 Third experiment neural diagram. At the left, the mobile robot with its six sensors (in green) is shown. At the right, the color grading output of the six neurons connected to the sensors is shown, representing different distances detected. Each of the input for the motor neurons are the six sensors of the robot. The output of each sensor is the input for an I&F neuron. These neurons produce spikes at a certain frequency depending on the distance it detects. These I&F neurons are connect by R-STDP to the output motors neurons that encode the direction to turn by the robot. The decision would be made depending on the spike count in the running time

change the behaviour of the network according to the environment. (2) The possibility of use different types of synapses, defined by the user. The system was implemented using the Izhikevich oscillator and Leaky Integrate and Fire (LIF) neurons.. The limitations of the current work are the resource consumption of the learning rule. The instantiation of an R-STDP module is made for each synapse. Thus, the update of the weights can be performed in parallel, taking advantage of the available resources in the FPGA. However, for large neural structures (+100 neurons) can be a limitation in terms of resource usage, as it increases with the number of synapses in the network.

4 Conclusions

R-STDP has demonstrated its use in both, robotics and computer vision fields. This paper presents a fully digital implementation of a learning method in an SNN using synaptic plasticity based on R-STDP. Izhikevich neurons are used together with a learning signal, representing the dopamine concentration found in the network. This signal allows us to modulate the learning of the network based on the conditions of the environment in which it is located.

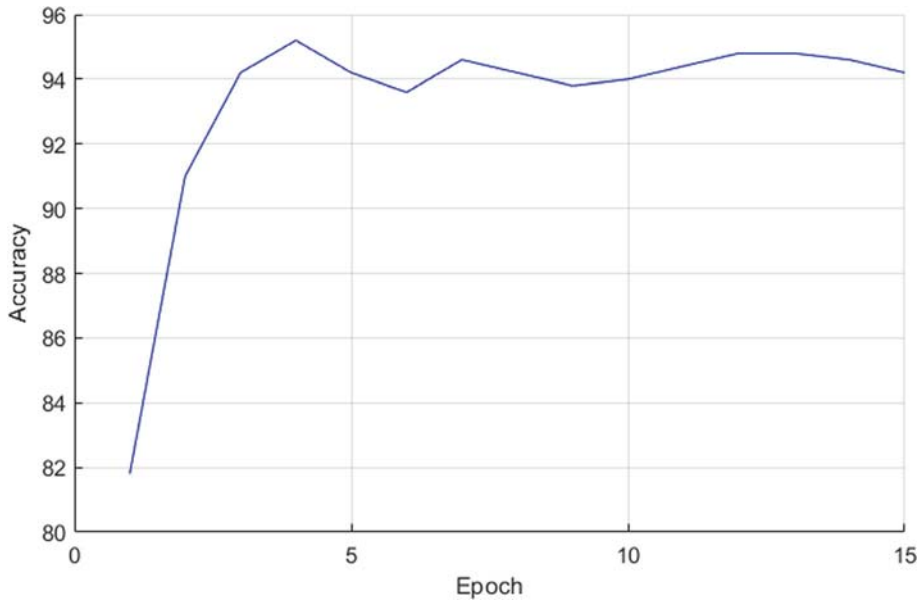

Fig. 10 Accuracy of the third experiment simulation over 15 epochs

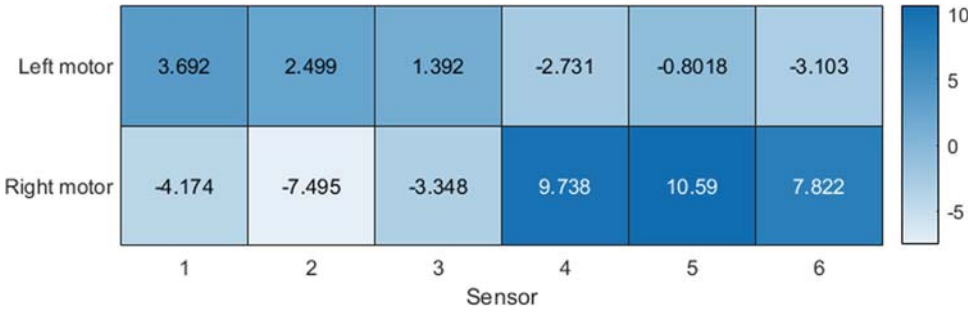

Fig. 11 Final weight distribution in the third experiment obtained on the FPGA. Each row represent a different motor neuron and each column a different ultrasonic sensor

Table 5 Comparison between proposed method and previously published methods.

Reference	LUTs	DSP	Neural model	Synapse model	External signal[a]	Training method
Ours[b]	394	1	Izhikevich, LIF	Current based[c]	yes	R-STDP
Heidarpur et al. [8][d]	410	0	Izhikevich	–	no	STDP
Cassidy et al. [5]	16[e]	–	–	–	no	STDP
Yousefzadeh et al. [19]	–	–	–	1-bit	no	Stochastic STDP
Hu SG et al. [9]	–	–	I&F	Conductance based 4-bit	no	STDP
Lammie et al. [14][f]	859	362	–	–	no	Pair-based STDP
	1943	362	–	–	no	Triplet-based STDP

[a]Using an external signal for learning, such as dopamine. This allows the possibility to use it as a supervised learning method, as demostrated on the third experiment in the results

[b]Using a memory

[c]Users can define the synapse dynamic (exponential, alpha, biexponential, etc.

[d]Area optimized version

[e]Comb. encoding III

[f]Digital full resolution version

The results obtained are comparable with those simulated by Brian2, showing a very accurate behaviour in a realtime platform, opening new perspectives to implementing in real time complex learning processes in spiking neural networks.

Acknowledgements Fernando M. Quintana would like to acknowledge the Spanish *Ministerio de Ciencia, Innovación y Universidades* for the support through FPU grant (FPU18/04321). This work was part of the the project Nemovision PID2019-109465RB-I00, supported by MICIU/AEI/ 10.13039/5011000110, the Spanish grant (with support from the European Regional Development Fund) MIND-ROB (PID2019-10555 6GBC33) and by the EU H2020 project CHIST-ERA SMALL (PCI2019-111841-2).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declaration

Conflict of interest We wish to confirm that there are no known conflicts of interest associated with this publication.

References

1. Bing Z, Meschede C, Huang K, et al (2018) End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle. In: IEEE international conference on robotics and automation. IEEE, pp 4725–4732. https://doi.org/10.1109/ICRA.2018.8460482
2. Bing Z, Baumann I, Jiang Z et al (2019) Supervised learning in snn via reward-modulated spike-timing-dependent plasticity for a target reaching vehicle. Front Neurorobot 13:18. https://doi.org/10.3389/fnbot.2019.00018
3. Bing Z, Jiang Z, Cheng L, et al (2019b) End to end learning of a multi-layered snn based on r-stdp for a target tracking snake-like robot. In: Proceedings–IEEE international conference on robotics and automation. Institute of electrical and electronics engineers Inc., pp 9645–9651. https://doi.org/10.1109/ICRA.2019.8793774
4. Buchanan K, Mellor J (2010) The activity requirements for spike timing-dependent plasticity in the hippocampus. Front Synaptic Neurosci 2:11. https://doi.org/10.3389/fnsyn.2010.00011
5. Cassidy A, Andreou AG, Georgiou J (2011) A combinational digital logic approach to stdp. In: IEEE international symposium on circuits and systems, pp 673–676. https://doi.org/10.1109/ISCAS.2011.5937655
6. Furber SB, Galluppi F, Temple S et al (2014) The spinnaker project. Proc IEEE 102:652–665. https://doi.org/10.1109/JPROC.2014.2304638
7. Gerstner W, Kistler WM, Naud R, et al (2014) Synaptic plasticity and learning. In: Neuronal dynamics: from single neurons to networks and models of cognition. Cambridge University Press, pp 491–523. https://doi.org/10.1017/CBO9781107447615.023
8. Heidarpur M, Ahmadi A, Ahmadi M et al (2019) Cordic-snn: on-fpga stdp learning with izhikevich neurons. IEEE Trans Circuit Syst I Regul Pap 66:2651–2661. https://doi.org/10.1109/TCSI.2019.2899356
9. Hu SG, Qiao GC, Chen TP et al (2021) Quantized stdp-based online-learning spiking neural network. Neural Comput Appl 2021:1–16. https://doi.org/10.1007/S00521-021-05832-Y
10. Humaidi AJ, Kadhim TM, Hasan S, et al (2020) A generic izhikevich-modelled FPGA-realized architecture: a case study of printed english letter recognition. In: Proceedings of 2020 24th international conference on system theory, control and computing, ICSTCC 2020. Institute of Electrical and Electronics

Engineers Inc., pp 825–830. https://doi.org/10.1109/ICSTCC50638.2020.9259707
11. Indiveri G, Linares-Barranco B, Hamilton T et al (2011) Neuromorphic silicon neuron circuits. Front Neurosci 5:73. https://doi.org/10.3389/fnins.2011.00073
12. Izhikevich EM (2003) Simple model of spiking neurons. IEEE Trans Neural Netw. https://doi.org/10.1109/TNN.2003.820440
13. Izhikevich EM (2007) Solving the distal reward problem through linkage of stdp and dopamine signaling. Cereb Cortex 17:2443–2452. https://doi.org/10.1093/cercor/bhl152
14. Lammie C, Hamilton TJ, van Schaik A et al (2019) Efficient fpga implementations of pair and triplet-based stdp for neuromorphic architectures. IEEE Trans Circuit Syst I Regul Pap 66:1558–1570. https://doi.org/10.1109/TCSI.2018.2881753
15. Moradi S, Qiao N, Stefanini F et al (2018) A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). IEEE Trans Biomed Circuit Syst 12:106–122. https://doi.org/10.1109/TBCAS.2017.2759700
16. Morrison A, Diesmann M, Gerstner W (2008) Phenomenological models of synaptic plasticity based on spike timing. Biol Cybern 98(6):459–478. https://doi.org/10.1007/s00422-008-0233-1
17. Stimberg M, Brette R, Goodman DF (2019) Brian 2, an intuitive and efficient neural simulator. eLife. https://doi.org/10.7554/eLife.47314
18. Vasquez Tieck JC, Becker P, Kaiser J, et al (2019) Learning target reaching motions with a robotic arm using brain-inspired dopamine modulated STDP. In: Proceedings of 2019 IEEE 18th international conference on cognitive informatics and cognitive computing, ICCI*CC 2019. Institute of Electrical and Electronics Engineers Inc., pp 54–61. https://doi.org/10.1109/ICCICC46617.2019.9146079
19. Yousefzadeh A, Stromatias E, Soto M et al (2018) On practical issues for stochastic stdp hardware with 1-bit synaptic weights. Front Neurosci 12:665. https://doi.org/10.3389/fnins.2018.00665