Low-Energy and Fast Spiking Neural Network For Context-Dependent Learning on FPGA

Hajar Asgari, Babak Mazloom-Nezhad Maybodi, Melika Payvand, and Mostafa Rahimi Azghadi

***Abstract*—Supervised, unsupervised, and reinforcement learning (RL) mechanisms are known as the most powerful learning paradigms empowering neuromorphic systems. These systems typically take advantage of unsupervised learning because they can learn the distribution of sensory information. However, to perform a task, not only is it important to have sensory information, but also it is required to have information about the context in which the system is operating. In this sense, reinforcement learning is very powerful for interacting with the environment while performing a context-dependent task. The predominant motivation for this research is to present a digital architecture for a spiking neural network (SNN) model with RL capability suitable for learning a context-dependent task. The proposed architecture is composed of hardware-friendly leaky integrate-and-firing (LIF) neurons and spike timing dependent plasticity (STDP)-based synapses implemented on a field programmable gate array (FPGA). Hardware synthesis and physical implementations show that the resulting circuits can faithfully reproduce the outcome of a learning task previously performed in both animal experimentation and computational modelings. Compared to the state-of-the-art neuromorphic FPGA circuits with context-dependent learning capability, our circuit fires 10.7 times fewer spikes, which accelerates learning 15 times, while requiring 16 times less energy. This is a significant step in achieving fast and low-energy SNNs with context-dependent learning ability on FPGAs.**

***Index Terms*—Neuromorphic engineering, field programmable gate array (FPGA), context-dependent task.**

I. Introduction

BECAUSE of highly efficient performance-resources trade-off of the biological brain in learning tasks, research into hardware realization of brain-inspired computing platforms is becoming increasingly popular [1]–[4]. Specifically, SNNs are gaining more attention because of their biological plausiblity and low power consumption as a result of the sparse activity of their neurons [5]. Many groups have worked on different implementation of such networks in analog and digital systems [1]–[3], [6], [7]. Such intelligent low power systems are perfect candidates for autonomous agents which can learn to interact with the environment in an unsupervised fashion [8]. However, the promise of the autonomous system is fundamentally based on the ability of the agent to learn and act not only based on the sensory information, but also based on the context and state it is operating at. Learning a context dependent task thus involves encoding and remembering the context of an event in which an item of object has been observed [9]. Performing such task requires a precise, flexible, and reliable implementation substrate such as FPGAs [10]. Several research groups have proposed approaches for modeling SNNs on FPGA platforms [10]–[13]. However, efficient digital implementation of SNNs with reinforcement learning is yet to be fully explored.

This paper proposes a digital hardware architecture for spiking neural networks with reinforcement learning. This network is able to reproduce hippocampus principal neurons behavior while learning a context-dependent task. Our novel hardware is significantly faster than its counterpart FPGA designs, while requiring much lower operational energy.

The study starts with a description of the context-dependant task and SNN model in Section II. In Section III, we introduce the hardware architectures for neurons, synapses, and network. Section IV presents the implementation results accompanied by an extensive comparison between the proposed and previous studies. Finally, section V concludes the paper.

II. Background

This paper presents a digital spiking neural network with reinforcement learning capability. To validate the proposed network operation in a real task, an SNN for learning a context-dependent task is designed using the proposed architecture. In this section a few concepts and SNN model of the context-dependent task are described.

A. Description of The Context-Dependent Task

This task was designed to probe the behavior of individual hippocampal principal neurons during learning rewarded items depending on the environmental context properties [14]. In the original experiment on rats, there were two boxes as contexts A and B with different wallpapers and floors and a shared entrance which separates the two boxes. Inside each of the boxes, two pots (Item X and Y) were randomly located in two different positions (position 1 and 2) and only one of them contained a reward. In the experiment, animals were trained to choose item X in context A and item Y in context B, regardless of which position the items were located [15].

B. Spiking Neural Network Model

All visual appearance and spatial positions are abstracted in the form of binary input vectors [14]. As shown in Fig. 1,

H. Asgari is a visiting Ph.D. student at Institute of Neuroinformatics (INI), Zurich, Switzerland from Shahid Beheshti University (SBU) of Tehran, Iran (e-mail: h_asgari@sbu.ac.ir).

B. Mazloom-Nezhad Maybodi is with the Department of Electrical Engineering, Shahid Beheshti University, Tehran, Iran (e-mail: b-mazloom@sbu.ac.ir).

M. Payvand is with the Institute of Neuroinformatics (INI), University of Zurich and ETH Zurich, Zurich, Switzerland (e-mail: melika@ini.uzh.ch)

M. Rahimi Azghadi is with the College of Science and Engineering, James Cook University, Townsville, QLD 8414, Australia (e-mail: mostafa.rahimiazghadi@jcu.edu.au)

the network for the context-dependent task includes an input (sensory) layer, a hidden (Hippocampal) layer and an output (motor) layer. In this model, the input layer includes six neurons (which represent two contexts, two items, and two positions) and provides eight different spatial information. Adaptive excitatory weights connect all these input neurons to all hippocampal neurons. All eight hippocampal neurons in the hidden layer have inhibitory connections among themselves, without self-inhibition. A few plastic excitatory synapses connect hippocampal neurons to neurons in the output layer. Similar to the hippocampus layer, neurons in the output layer have inhibitory connections within themselves. As the output layer includes two neurons, there are two actions in the network output: dig and move[1]. Hidden and output layers are two separate winner-take-all (WTA) networks [14].

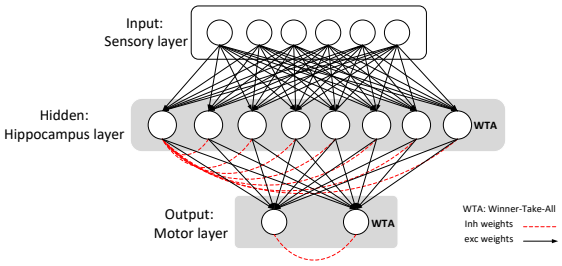

Fig. 1: The spiking neural network, used to model the reinforcement learning in the context-dependent task. All to all excitatory synapses connect layers in this network. Inhibitory synaptic connections among all neurons provide WTA networks. Only inhibitory synapses of one neuron are shown.

III. Hardware Design and Implementation

During our investigations at the design stage, we found out that replicating the biological and computation results for the targeted task require high-precision calculations. Therefore, 32-bit number representation was chosen, which enables us to achieve state-of-the-art results. With the aim of achieving the maximum computational efficiency and a minimal hardware cost, here we incorporate approximation techniques for neurons and plastic synapses. Fig. 2 demonstrates our neuromorphic system composed of five blocks. Here, similar to all other neuromorphic systems `Neurons Block` and `Synapses Block` are the main parts of the proposed network. All connections among neurons and synapses are defined in `Crossbar` block. Competition among neurons groups including WTAs networks are done in `Activities plus WTAs` block. The `Peripheral Block` contains all the other required cores for synapse initializing and controlling network sequences. The architecture of each block and system operation for the context-dependent task are described in the following subsections.

A. Neurons Block

In this block all neurons are LIF [16] and are sorted in a column. Direct discretization of neuron's dynamic results in Eq. 1 with a constant time step (dt). The membrane voltage V_m is the potential of capacitance C that is driven by input current I while there is leakage current through G_l channel. In this equation, V_{reset} is the resting potential for the membrane. Additionally when membrane voltage crosses a threshold voltage (V_{th}) and goes above it, the neuron fires and V_m sets to V_{reset} [16]. However, in this model for calculating membrane voltage two multiplication operations are required which results in high area and power consumption.

$$V_m[n+1] = (1 + \frac{G_l}{C}dt)V_m[n] + \frac{dt}{C}I[n] - \frac{G_l}{C}dtV_{reset} \quad (1)$$

If C, G_l and dt values are chosen so that $V_m[n] \gg \frac{G_l}{C}dtV_m[n]$, Eq. 2 can be efficiently approximated to:

$$V_m[n+1] \simeq V_m[n] + V[n] - V_{leakage} \quad (2)$$

where $V[n] = \frac{dt}{C}I[n]$ is the input voltage and $V_{leakage} = \frac{G_l}{C}dtV_{reset}$ is leakage potential.
As shown in Fig. 3(a) this hardware is only composed of add and accumulation operations without any multipliers. This introduces a significant cost reduction in the neuron block.

B. Synapses Block

The other main unit of the network is the synapse. In these networks, there are two synapse types: Inhibitory static synapses and excitatory plastic synapses. The first group of synapses has the strength of W=1 and provides strong inhibition. In contrast to inhibitory synapses, the excitatory synapses enhance neurons membrane voltage. The values of these synapses are modified during the replay phase based on an STDP-based learning mechanism. In the STDP algorithm, the weight of a synapse modifies according to the timing difference between the pre- and post-synaptic spikes arrival. As shown in Fig. 4(a) the amount of modification in the learning rule implemented in [14] depends on both arrival timing difference and the strength of synapse just before adaptation. In our implementation, we have utilized look-up-table (LUT) approximation which is an efficient technique

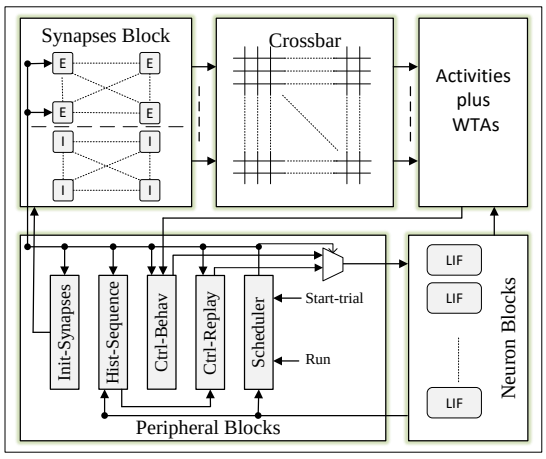

Fig. 2: High level block diagram of the network architecture.

[1]"Dig" in the experiment is an action that the agent takes if it decides that there is a reward hidden in the current selection. Otherwise, the agent walks to a different location, an action termed "move" in the experiments.

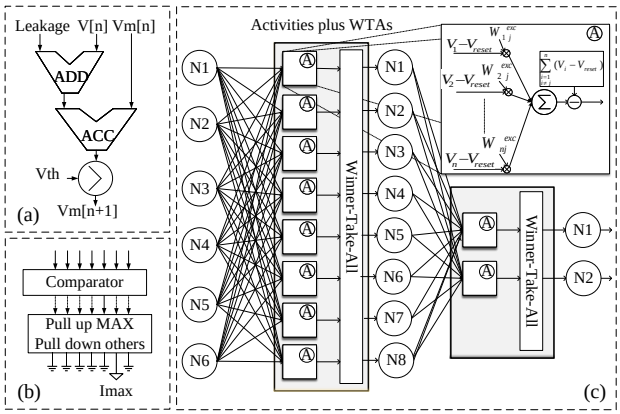

Fig. 3: (a) Simplified LIF neuron model. (b) Winner-Take-All block diagram. (c) Low-level block diagram for calculating neurons activity and WTA networks.

for reducing implementation complexities of neuromorphic designs [17]. Fig. 4(b) shows the comparison between the LUT-based synapse model with the size of 16×10 and the original model for various spike time differences. In our implementation, the initial values for excitatory weights are small random numbers between 0 and 1.

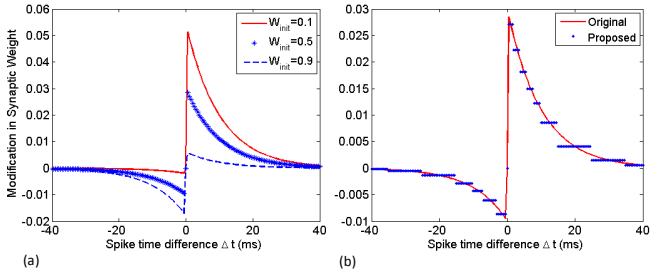

Fig. 4: (a) Synaptic modification over spike timing difference for three various initial weights [14]. (b) LUT-based approximation of the synapse model ($W_{init} = 0.5$).

C. Error Analysis

As a consequence of the estimations for neuron and synapse dynamics here, we use root means square error (RMSE) to measure the time domain error of the approximated system compared to the original model. The error equation is defined as in [10]. In addition, similar to the previous studies, normalized RMSE (NRMSE) for neuron and synapse approximations are calculated (shown in Table I) to evaluate the error in our approximate models compared to the original model simulations. Based on the small estimated errors, both proposed models for synapse and neuron comparatively replicate original neuron and synapse behavior very closely.

D. Crossbar and Activities plus WTAs block

Crossbar block specifies all connections among neurons and synapses. Hippocampal and output layers contain WTA networks. As shown in Fig. 3(b) each typical WTA is composed

TABLE I: Error values calculated for our approximate implementations, compared to other similar models.

	Proposed	Proposed	[10]	[4]
Rule	Neuron	Synapse	Neuron	Synapse
RMSE (%)	0.4966	0.155	3.49	-
NRMSE (%)	7.0942	2.146	3.53	5.9

of a comparator and an output assignment unit which pulls up neurons input with the highest activity and pulls down others. Fig. 3(c) illustrates a low-level block diagram of how WTA networks are located in this SNN architecture with more details. Blocks A in Fig. 3(c) calculate the activity of cells (A^{cell}) using Eq. 3.

$$A_j^{cell} = \sum_{i=1}^{i=n_k} (V_i - V_{reset}) W_{ij}^{exc} - \sum_{i=1, i\neq j}^{i=n_l} (V_i - V_{reset}) W_{ij}^{inh} \tag{3}$$

where i and j are the indexes of the neurons in the pre- and post-synaptic layers, respectively. n_k and n_l are total neurons in these layers. After calculating the cells activity, a WTA in each layer compares all values to make sure that in each layer only neurons with the highest activity are pulled up.

E. Peripheral Blocks

As shown in Fig. 2, this block contains all the other sub-blocks responsible for controlling the network behavior. `Scheduler` manages sequences in this network. This unit gets neurons spikes and several controlling signals and based on them provides controlling signals for the other units. `Init-Synapses` gets orders from the scheduler and provides initial values for synaptic weights. This core includes several Linear Feedback Shift Register (LFSR) block to generate random numbers for each synaptic weight. The `Ctrl-Behav` unit gets WTAs outputs and a controlling signal from the scheduler and then provides input voltage for neurons during the behavioral phase. `Ctrl-Replay` unit gets controlling signals from the scheduler and provides input voltage for neurons during the replay phase to replicate the firing patterns of the neurons during the feed-forward operation which is used for the learning procedure. `Hist-Sequence` unit stores neurons states (active or inactive) during the behavioral phase and provides the required information for the `Ctrl-Replay` unit during the replay phase.

F. Analysis of the Training Mechanism

After getting a `Run` order, all initial values for both inhibitory and excitatory synapses are set. By arriving `Start-trial` order, the behavioral phase starts and neurons firing continue until getting a "dig" (with or without reward) from the output layer. Dig event causes the network to switch to the replay phase, in which the scheduler makes all plastic synapses eligible to be modified. In case of getting a reward signal, two latest stored action-sequences replay in forward direction as is explained in section III-E. This way of replaying causes related synaptic weights to enhance. Otherwise, the network will replay in the reverse direction so several synapses will be depressed. This procedure repeats around 100 times, after which the network has learned the task.

IV. Hardware Results and Discussion

The proposed system contains LIF Neurons and STDP-based synapses with reinforcement learning adaptation mechanism. Our designed network for the context-dependent task contains 16 neurons, 64 excitatory plastic synapses, and 58 inhibitory fixed synapses. All neurons, synapses and other peripheral parts are described using the standard top-bottom digital design flow. In order to minimize FPGA resource utilization, neuron and synapse blocks are designed efficiently (as described in Section III-A and B). As a proof of concept we implement our learning network on Kintex-7 XC7kt160t FPGA which is hosted in opal kelly XEM7360 board. This device contains 202800, 101400, and 600 Slice FFs, Slice LUTs, and DSPs, respectively. Based on the results from Table II, the entire proposed network uses only 2.51 percent of available Slice FFs, 34.17 percent of available Slice LUTs and 42.67 percent of available DSPs. This table also reports the total on-chip power and energy based on the analytic of the Xilinx Power Estimator (XPE) after HDL synthesis. Moreover, Table II reports network latency which is defined as the maximum delay time for making a decision in a trained network. Table II also compares our proposed SNN to several other studies which implement SNNs on FPGA for different tasks. Please note that none of these FPGA networks except [18], are capable of context-dependent on-line learning. Furthermore, FPGA devices and synthesizer versions that have been used for implementation are different. Therefore the device utilization results presented in these tables must be considered relatively.

We run the proposed network with randomly initialized synaptic weights using Xilinx Vivado Design suit. During the training of the network, all the results were stored in different files and then analyzed and plotted in MATLAB. Fig. 5 depicts the network performance in a number of trials. Performance is calculated by measuring the percentage of the mean number of correct responses over a sliding window of 30 trials. The proposed network is tested for more than 30 runs with different initial synaptic weights. The figure shows that our hardware is able to learn the task within 100 trials and reaches an approximate 90 % mean performance similar to the computational model [14]. In addition, in the animal experiment the context-dependant task was learned in about 100 trials and performance reached about 80% to 90% correct behavioral responses [15]. Thus, the proposed digital network can faithfully model the animal experiment result. Fig. 6 illustrates total number of spikes over each input triplet. This figure and raster plots (see supplementary material) confirm that neurons activities in the proposed network are similar to the computational model. Fig. 6 shows four out of eight neurons in hippocampus layer mostly participate in functional network. To predict the reward, this network makes a link between item and context to be associated with reward, independent of items position. Toward the end of the simulation, neuron 1 fires selectively for item Y in context B, regardless of the item Y place. Neuron 2 fires selectively for item Y in context A. Neuron 5 fires selectively for item X in context A. As it can be seen, neuron 5 also fires for B2X. This is because, at the beginning of the trials for input triplets B1X and B2X, different neurons fire. However, after learning is complete, Neuron 6 is trained to fire selectively for item X in context B regardless of its position. It is worth noting that, all the observed firing patterns are in good agreement with animal experiments [15] and computational models [14] and confirm the functionality of our developed network on FPGA.

In a previous study performed on a digital substrate with the same capabilities, we developed a network requiring less FPGA resources [18]. However, the neurons in that network required a large firing rate for learning, which resulted in high energy consumption. Besides, tuning the network parameters proved much more time-consuming [18]. Neurons' membrane voltages in the previous architecture are dependent on synaptic weights strengths. To increase network learning memory, weight adaptation parameters should be chosen as small values. So for a successful learning of a task, the previous network needs many repetitions. These parameters also affect the operation of WTA networks. The contribution of this work is to reduce this dependency by employing "Activities plus WTAs" (shown in gray box in Fig. 3) which results in faster convergence and lower firing rate (energy consumption).

TABLE II: Comparison of our developed neural network with a number of previous spiking networks implemented on FPGAs. Abbreviations: Slice FFs (S.FFs), Slice LUTs (S.LUTs), Max Frequency (M.F), This Work (T.W), Virtex (V), Spartan (S), Kintex (K), Off-line learning (OFL), and Online-learning (ONL).

	[10]	[13]	[11]	[12]	[18]	T.W
S.FFs	1023	50228	-	1676	8906	5088
S.LUTs	11339	86032	-	6214	19059	34646
DSPs	0	1112	-	32	0	256
M.F (MHz)	189	63.389	75	25	148.4	143
Device	V.6	V.7	S.6	S.6	K.7	K.7
Power (W)	-	-	1.5	-	1.81	1.91
Energy (μJ)	-	-	-	-	2257	139.5
Learning	OFL	-	OFL	OFL	ONL	ONL
Latency (μs)	-	-	-	-	-	45.6

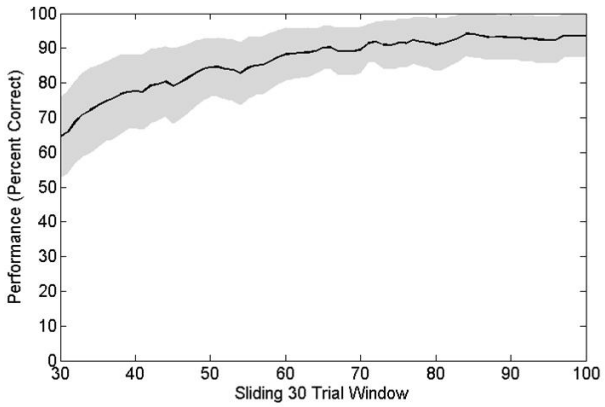

Fig. 5: Behavioral performance during successful learning of the context-dependent task for more than 30 runs (with different initial weights). The solid line shows the mean and the gray area behind the solid run shows the standard deviation from the mean. Notice that the first 30 trials are not shown because of boundary effects caused by using the sliding window of 30 trials.

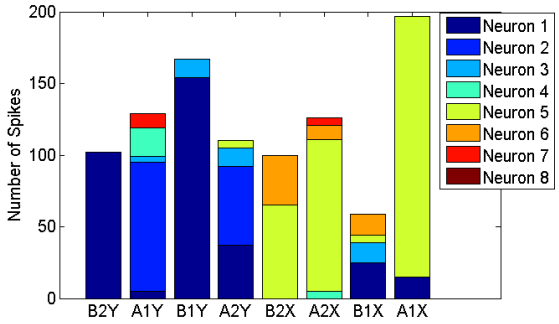

Fig. 6: Number of neurons spikes over each triplet during network operation in a successful learning of the context-dependant task.

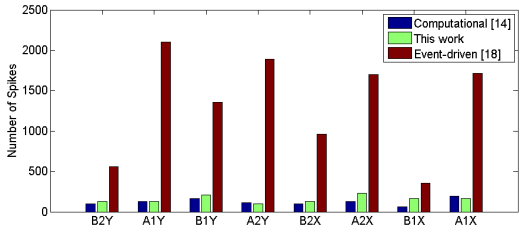

Fig. 7: Comparison of total number of spikes of our proposed network, with computational simulations [14] and our previous event-driven study [18].

Activities plus WTAs block is responsible for finding neurons with a higher activity rate. Fig. 7 shows a comparison of the total number of spikes in each learning triplet for the proposed network, computational simulations [14], and our previous study [18]. As shown, the proposed network has a firing rate close to the computational model and significantly lower than the prior event-driven hardware architecture. The total number of spikes in a successful learning run and the mean required times for each learning trial for the computation simulations that run on a core i7 processor (1.6 GHz), as well as the event-driven [18] and the proposed architectures on FPGA at f_{clk} = 100 MHz, are presented in Table III. As shown, the proposed architecture is 15 and 37 times faster than the previous FPGA counterpart, and computational simulations, respectively. Additionally, as Table III shows, our proposed hardware requires to generate 10.7 times fewer spikes, while learning the same task learned by the event-driven hardware proposed in [18]. This smaller number of spikes and the shorter time required for learning, leads to a significant 16-fold energy saving, as reported in Table II. These make our new design suitable for learning context-dependent tasks on FPGAs, which can improve real-time reinforcement learning in various applications including robotics.

TABLE III: The total number of spikes in a successful learning run and mean elapsed time for each learning trial.

	Computational	Event-driven	This Work
Total number of spikes	1251	10633	990
Mean elapsed time	2.85 s	1.18 ms	77 μs

V. Conclusion

This paper presents an implementation of a context dependent task on an FPGA using a spiking neural network with reinforcement learning capability. Neurons and synapses are designed to minimize hardware cost. The proposed network has been synthesized and as a proof of concept implemented on Xilinx Kintex-7 FPGA device. The implemented hardware shows performance on par with the computational model and also animal experiment of previous studies. Moreover, results show that the proposed network model has a significantly lower firing rate and energy consumption and converges faster in comparison with previous hardware architecture while learning the same task. The result of this paper can facilitate research to employ RL using neuromorphic systems.

References

[1] N. Qiao *et al.*, "A Reconfigurable On-Line Learning Spiking Neuromorphic Processor Comprising 256 Neurons and 128K Synapses," *frontiers in Neuroscience*, vol. 9, no. 141, pp. 1–17, 2015.

[2] F. Akopyan *et al.*, "TrueNorth: Design and Tool Flow of a 65mW 1 Million Neuron Programmable Neurosynaptic Chip," *IEEE Trans. Computer-Aided Design of Integrated Syst*, vol. 34, no. 10, pp. 1537–1557, 2015.

[3] C. Frenkel *et al.*, "A 0.086-mm2 12.7-pJ/SOP 64k-Synapse256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28nm CMOS System," *IEEE Trans. Biomedical Syst*, vol. 13, no. 1, pp. 145–158, 2019.

[4] C. Lammie *et al.*, "Efficient FPGA Implementations of Pair and Triplet-Based STDP for Neuromorphic Architectures," *IEEE Trans. Circuits Syst. I*, vol. 66, no. 4, pp. 1558–1570, 2019.

[5] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons," *IEEE Trans. Neural Netw*, vol. 15, no. 5, pp. 1063–1070, 2004.

[6] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE Trans. Biomedical Syst*, vol. 12, no. 1, pp. 106 – 122, 2018.

[7] M. Davies *et al.*, "Loihi: A Neuromorphic Many Core Processor With On-chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[8] M. B. Milde *et al.*, "Obstacle Avoidance and Target Acquisition For Robot Navigation Using A Mixed Signal Analog/digital Neuromorphic Processing System," *Frontiers in Neurorobotics*, vol. 28, p. 11, 2017.

[9] E. Neftci *et al.*, "Synthesizing Cognition In Neuromorphic Electronic Systems," *Proceedings of the National Academy of Sciences*, vol. 110, no. 37, pp. E3468–E3476, 2013.

[10] E. Z. Farsa *et al.*, "A Low-Cost High-Speed Neuromorphic Hardware Based on Spiking Neural Network," *IEEE Trans. Circuits Syst. II*, vol. 66, no. 9, 2019, doi: 10.1109/TCSII.2019.2890846.

[11] D. Neil and S. Liu, "Minitaur, an Event-Driven FPGA-Based Spiking Network Accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2621–2628, Dec 2014.

[12] D. Ma *et al.*, "Darwin: A Neuromorphic Hardware Co-processor Based on Spiking Neural Networks," *Journal of Computational Electronics*, vol. 77, pp. 43–51, 2017.

[13] S. Y. Bonabi *et al.*, "FPGA Implementation of A Biological Neural Network Based on The Hodgkin-Huxley Neuron Model," *Frontiers in Neuroscience*, vol. 8, no. 379, pp. 1–12, 2014.

[14] F. Raudies and M. E. Hasselmo, "A Model of Hippocampal Spiking Responses to Items During Learning of a Context-Dependent Task," *frontiers in System Neuroscience*, vol. 23, no. 8, pp. 1–12, 2014.

[15] R. W. Komorowski *et al.*, "Robust Conjunctive Item-place Coding by Hippocampal Neurons Parallels Learning What Happens Where," *Neural Comput.*, vol. 29, no. 31, pp. 9918–9929, 2009.

[16] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[17] F. Galluppi *et al.*, "A Framework for Plasticity Implementation on the SpiNNaker Neural Architecture," *Frontiers in Neuroscience*, vol. 8, no. 429, pp. 1–20, 2015.

[18] H. Asgari *et al.*, "Digital Multiplier-less Event-Driven Spiking Neural Network Architecture for Learning a Context-Dependent Task," *arXiv:1906.09835*, 2019.