An Efficient Hardware Architecture for Multilayer Spiking Neural Networks

Yuling Luo[1], Lei Wan[1], Junxiu Liu[1]✉, Jinlei Zhang[1], and Yi Cao[2]

[1] Guangxi Key Lab of Multi-source Information Mining & Security,
Faculty of Electronic Engineering, Guangxi Normal University, Guilin, China, 541004
[2] Department of Business Transformation and Sustainable Enterprise,
Surrey Business School, University of Surrey, Surrey, UK, GU2 7XH
liujunxiu@mailbox.gxnu.edu.cn

Abstract. Spiking Neural Network (SNN) is the most recent computational model that can emulate the behaviors of biological neuron system. This paper highlights and discusses an efficient hardware architecture for the hardware SNNs, which includes a layer-level tile architecture (LTA) for the neurons and synapses, and a novel routing architecture (NRA) for the interconnections between the neuron nodes. In addition, a visualization performance monitoring platform is designed, which is used as functional verification and performance monitoring for the SNN hardware system. Experimental results demonstrate that the proposed architecture is feasible and capable of scaling to large hardware multilayer SNNs.

Keywords: Spiking neural networks, hardware architecture, FPGA.

1 Introduction

Recently, Spiking Neural Network (SNN) is being studied due to its computational capabilities for solving problems. The researchers attempt to mimic this efficiency and build artificial neural systems in hardware device. However, one main challenge is that it is the computational complexity, area demanding, nonlinear operators and highly dense interconnection schemes, which limit the system scalability. For example, a SNN system normally includes a significant number of neurons and synapses and the complexity of inter-neuron connectivity increases significantly with the number of neurons and synapses. Recently, FPGA-based architectures are popular for the SNN implementations. For instance, a FPGA-based approach was proposed in [1] to provide biologically compatible neural networks and can simulate up to 100 neurons. Another 20×20 topology that used simplified Hodgkin-Huxley neuron model was proposed in [2]. Reducing the precision of the data helps to minimize the consumed hardware area, e.g. the approach of [3] can accommodate 64K neurons. Its membrane potential value is updated by a synaptic weight with a 5-bit width. For the inter-neuron connections, the networks-on-chip (NoC) has been mostly used

as an efficient SNN interconnect strategy. For example, a routing architecture based on two-dimensional (2D) mesh topology was proposed in the approach of [4]. The FACETS in the approach of [5] was based on a 2D torus which provided the connections of several FACETS wafers. A state-of-the-art hierarchical NoC architecture for SNN was proposed in the approach of [6], which combined the mesh and star topologies for different layers of the SNNs. Similar to the aforementioned approaches, most of current SNN hardware systems use the topologies with limited degree (e.g. mesh, torus etc.) to connect the neurons together, however these topologies do not meet the fan-in/out requirement of layer-based SNN structures [7]. Therefore, it is necessary to explore a new hardware architecture to address these challenges.

This paper presents a FPGA-based LTA architecture for the SNN neurons and synapses. The LTA employs a two-level sharing mechanism of computing components at the synapse and neuron levels and achieves a trade-off between computational complexity and hardware resource costs. For the connection problem between neurons, a compact and efficient routing architecture is proposed. It employs a traffic status weight-based arbitration and a traffic congestion-avoidance mechanism to provide traffic balance for the information transmission of multiple neurons. Further, a visualization performance monitoring platform is designed for the functional verification and performance monitoring of the hardware SNN.

2 SNN Hardware Architecture

The network topologies of the artificial neural networks can be divided into two categories, namely feed-forward and recurrent networks [8]. The feed-forward topology is mostly used and usually includes an input/output layer and one (or several) hidden layers. All the information is passed forward in one direction only. For example, a typical multilayer feed-forward network is shown in Fig. 1(a). Note that each neuron in a layer is connected to all neurons in the next layer by a weighted connection. The proposed LTA is illustrated in Fig. 1(b).

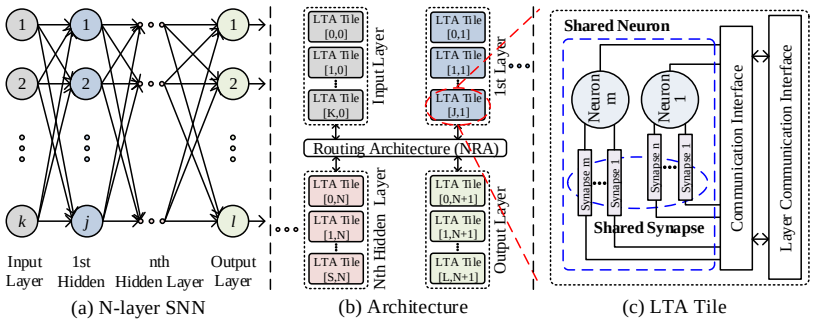


Fig. 1. An N-layer SNN and the overview of the SNN system using LTAs and NRA.

Each LTA corresponds to a layer in the SNN system in Fig. 1(a), i.e., it is programmed to realize a layer-level function of the SNNs. Inside the LTA, multiple neurons and synapses share computing components to provide a trade-off between computational complexity and the required hardware resources. For the LTAs in different layers, a novel routing architecture (NRA) is used as their communication infrastructure. The proposed LTA and NRA are presented in details in the following sections.

2.1 Compact Neuron Node

The micro-architecture of the proposed LTA is shown in Fig. 2. Based on the dynamic synapses model from the approach of [9], a digital synapse with sufficient resolution is developed and synthesized to an IP core as illustrated in the I/O block diagram of Fig. 2(a). It includes spike and configuration information inputs for the synapses (i.e., Spike_In and Config_In), the presynaptic and postsynaptic currents (i.e., I_In and I_Out), the variables (i.e., x, y, z_Excit_Inhib_In/Out, Use_In/Out) and the handshaking signals (i.e., hs_In and hs_Out). It uses floating-point data precision for the membrane potential value, weight etc. The digital synapse is used for synaptic computation and is shared by multiple synapses within a neuron cell. Using the digital synapse as the core component, the neuron architecture based on a typical Leaky Integrate-and-Fire (LIF) neuron model [10] is designed to model each neuron cell as shown in Fig. 2(b). The main components of a neuron (e.g., the soma, axon and synapse) are modularized, e.g. the Neuron Computing Core, the Cell Controller and the Communication Interface module implement the functions of the synapses, soma and axon, respectively. When packets are received from the Communication Interface, the Packet Decoder decodes them. For example, if the received data is

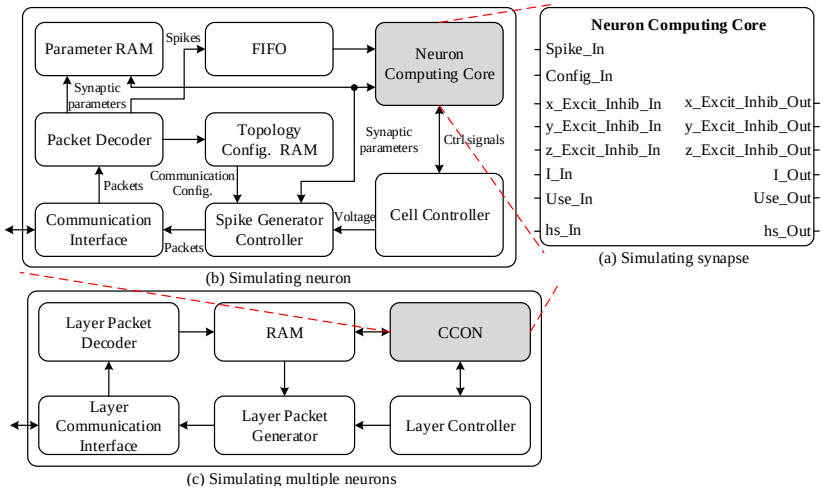


Fig. 2. LTA micro-architecture.

intended for configuration, then the parameters (e.g., the weight, decay, threshold value, lookup table and topology definitions) are saved in RAM for the synaptic computing and spike generation. When the spike packets arrive, the parameters and the initial membrane potential value are read from the Parameter RAM. Then, the NCC calculates the excitatory or inhibitory postsynaptic potentials of each synapse. The Cell Controller is used to manage the computation process. After completing all synapse computations, the Spike Generator Controller generates spike packets according to the communication parameters from the Topology Configuration RAM. In the final, it outputs the spike packets to the Communication Interface for the transmissions.

In this approach, in addition to the shared mechanism inside the neuron cell, neurons within the same layer (i.e., a single LTA) also share the computing component of one neuron (CCON) to further extend the efficiency of the design. Fig. 2(c) shows the proposed LTA diagram. The sharing mechanism of the LTA at the neuron level is similar to the sharing achieved the single neuron cell in Fig. 2(b). The CCON is the core module. The different neurons within an LTA share a single CCON. The RAM module is used to store the variables of different neurons. The Layer Controller manages the working flow of the LTA at the layer level. The Layer Packet Generator module is used to manage the data generation of different neurons.

2.2 Efficient Routing Architecture for the Inter-neuron Connection

In this work, an efficient interconnection architecture, i.e. NRA, is proposed, which can efficiently forward the spike events for the communications of multilayer SNNs. An example of interconnection between the NRA and LTA neuron nodes is introduced in Fig. 3(a). The NRA is based on an all-to-all connection that takes the paired input and output nodes of multilayer SNNs as the

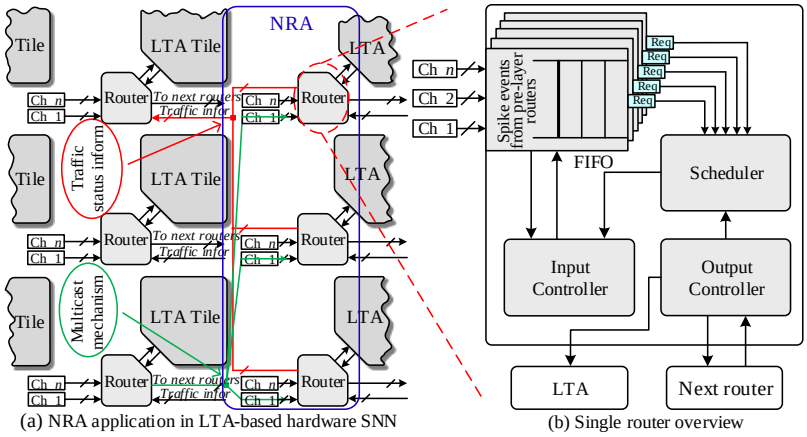


Fig. 3. The NRA application in multilayer SNN and its single router structure overview.

source and destination. A novel multicast-based NoC router is developed as the fundamental work unit of NRA and its structure is shown by Fig. 3(b). Each router has n input channels that receive the outputs from previous layer. And it fully connects to all the routes of the next layer. The traffic status information (e.g. busy or congested) are transmitted through the red line between different routers. After the LTA generates spike packets, the associated router forwards these packets to the routers in next layer. Considering that the NRA applies to multiple similar routers, only the structure and functionality of a single router are presented in details in the following content.

The single router diagram in Fig. 4(a) includes FIFO, Scheduler, Input controller, Output controller. Fig. 4(b) illustrates the hardware structure of the Scheduler. The structure of one traffic status weight computing unit is shown in Fig. 4(c). Five traffic status weight generators calculate the corresponding traffic status weights in real-time according to the input channel traffic status, which can be achieved by reading the statuses of corresponding FIFOs and grant information. The five traffic status weight signals including Present (w_p), Busy (w_b),

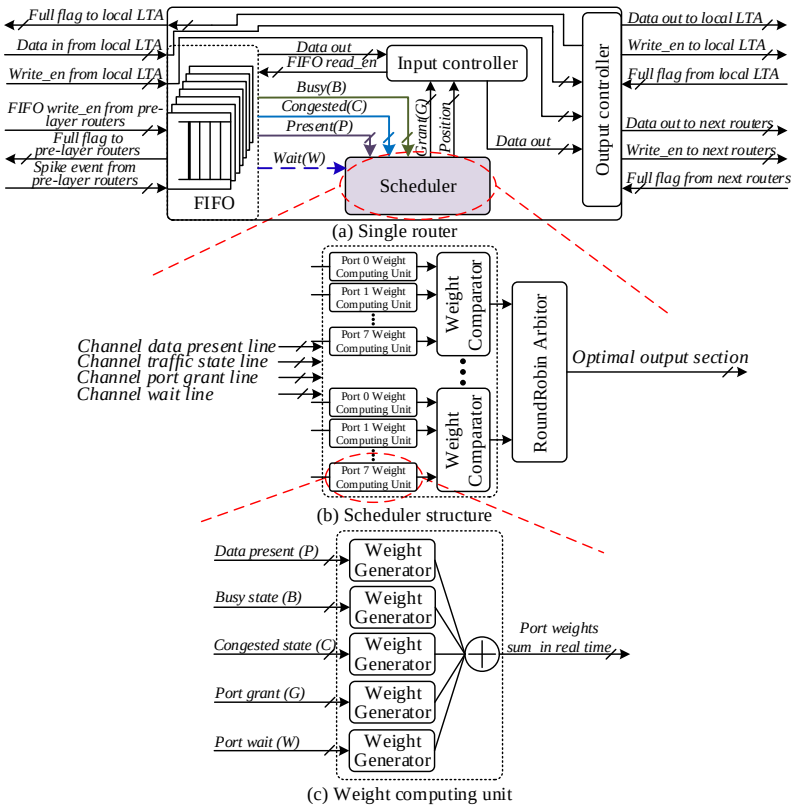


Fig. 4. The single router structure: (a) an overview of the single router; (b) the scheduler module; (c) the traffic status weight computing unit.

Congested (w_c), Grant (w_g) and Wait (w_w) are connected to the scheduler to aid making effective arbitration decisions and are calculated based on the following dynamic traffic status weight mechanism. If a channel has a data present, then the associated weight w_p is 3. Similarly, the weights for busy, congested, granted and waiting status are set to be 1, 2, -1 and 1, respectively. After all the traffic status weight values are generated, the total traffic status weight, w_{sum}, for each port i in each group, is calculated by $w_{sum}[i] = w_p[i]+w_b[i]+w_c[i]+w_g[i]+w_w[i]$ in real-time. The port with the largest traffic status weight in each group is selected to be granted by the round-robin arbiter. It can be noticed that the traffic status weight of each port is calculated in real-time. This allows the priorities of all ports are updated in real-time to access to the output channel [4].

2.3 A Visualization Performance Monitoring Platform for SNNs

Functional verification and performance monitoring of SNN hardware structure are challenges due to the SNN system complexity and large logic size. In this work, a visualization performance monitoring platform is designed. Fig. 5 illustrates the structure of this platform. The monitoring target is the Xilinx Zynq 7000 device. Its internal structure is divided into two parts of processor system (PS) and programmable logic (PL), i.e. a single chip integrates FPGA devices and dual-core ARM Cortex-A9 processing system. The high-speed AXI bus is used as the intercommunication between ARM and FPGA. The SNN hardware system is implemented in the FPGA device (Fig. 5 shows a typical network based on a 2D mesh topology). Considering the system lightweight design requirement, the monitoring platform uses the ARM processor to manage the workflow of the entire platform, signal statistics and communication. It connects to a computer via the high speed Ethernet communication. Only a compact signal collection

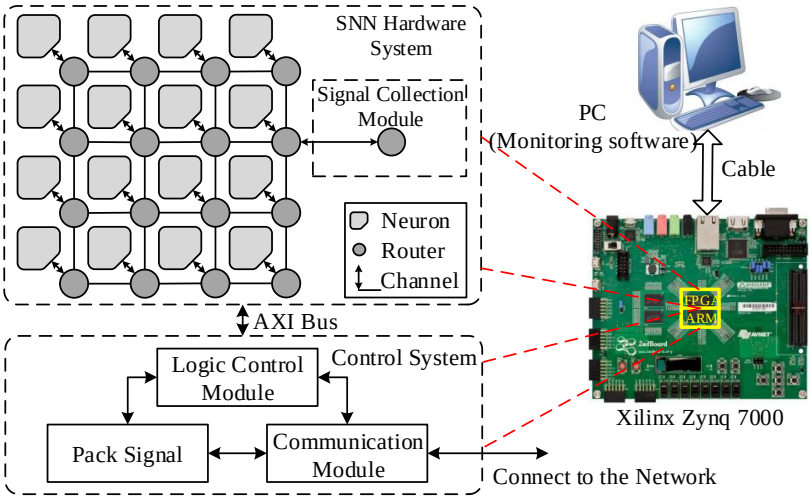


Fig. 5. The performance monitoring platform system overview.

unit is required for the PL side, which minimizes the resource overhead of the monitoring platform.

The computer software interface provides basic signal and data observation and control interfaces for each sub-module. It is divided into: (a) configuration module; (b) waveform display module; (c) frequency display module and (d) data display and storage module, which are shown by Fig. 6.

3 Experimental Results

3.1 Functional Evaluation

The XOR problem is used as a benchmark of functional evaluation in this experiment because it is a standard benchmark for verifying and evaluating artificially intelligent systems. A 2-layer feed-forward SNN consisting of 3 neurons (2 inputs and 1 output) was created to solve the XOR (see Fig. 7(a)). In this system, only 2 LTAs are required, one is used to model two neurons and four synapses in the

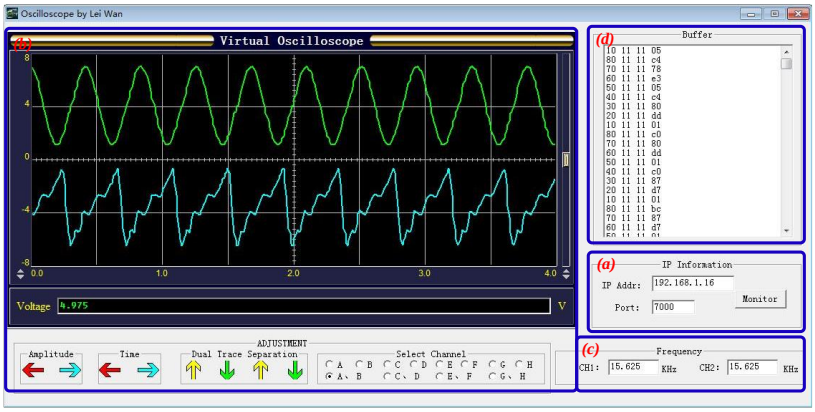
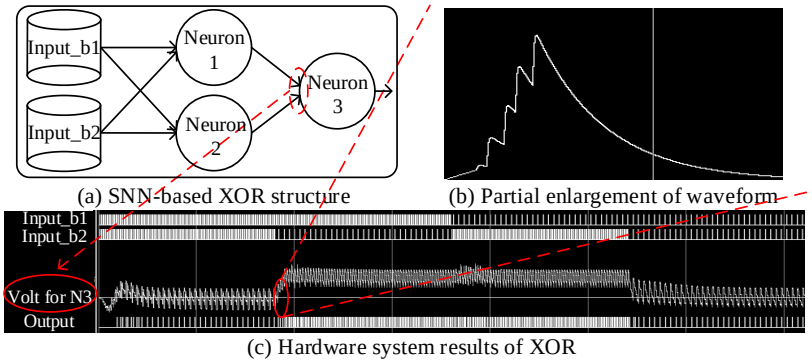


Fig. 6. The software interactive interface.



(a) SNN-based XOR structure (b) Partial enlargement of waveform

(c) Hardware system results of XOR

Fig. 7. Simulation results of XOR using the proposed LTA and NRA.

first layer network, another is used for one neuron and two synapses in the second layer. They are interconnected by the proposed routing architecture NRA. All the LTAs use the packets to transmit the information [4]. The rate encoding scheme is used in this paper, and the SNN can be trained using the learning algorithm in previous work [11]. The Xilinx Zynq-7000 development board (with a XC7Z020-CLG484 FPGA device) is used as the hardware platform. The results are shown in Fig. 7(c). The logical values '1' and '0' are encoded as spikes of 60 and 15 Hz, respectively. As expected, the output is '1' (i.e., 60 Hz) when both inputs are different and '0' (i.e., 15 Hz) when the inputs are the same. The membrane potential for neuron #3 is also shown in Fig. 7(b).

For large size networks, the IRIS and Wisconsin breast cancer (WBC) classification tasks can be used to analyse the scalability. The former is a 16:208:3 (i.e. 16 input, 208 hidden, and 3 output neurons) three-layer network, and the latter is 9:90:2 [6]. The IRIS or WBC requires only three LTAs and NRAs as each LTA can contain multiple neurons and a single NRA can provide communications for multiple LTAs. This demonstrates the scalability of the proposed architectures in this paper.

3.2 Performance analysis

Table 1 summarizes the recent approaches in terms of the network size, precision etc. The approaches in [3,8] used simplified LIF and SRM models and simplified routing architecture, which require relatively low hardware resources (e.g. the approach in [3] used only adders and subtractors for the synaptic computation and simplified 2D mesh-based NoC router for communication). Therefore, they can accommodate more neurons and synapses. The GPU-based approach [12] can simulate Izhikevich models by accelerating the computing process; however, it has the drawback of high power consumption. Our work can accommodate a

Table 1. Performance comparison with other approaches.

Approach	Network size	Precision	Power	Architecture	Device
[3]	64 K neurons 1.5 M synapses	5 bit	N/A	Modular neural tile	Virtex-6 200 MHz
[8]	50 neurons 1000 synapses	16-bit FXP	N/A	Mixed serial-parallel	Spartan3 69 MHz
[12]	55,000 neurons	32-bit FLP	∼7.35 W	Parallel	GPU 30 cores
[2]	400 neurons	32-bit FLP	N/A	20×20 topology	Virtex-4 100 MHz
[1]	100 neurons	32-bit FLP	N/A	Multiplexed neuron module	Virtex-6 100 MHz
This Work	1,728 neurons 43,200 synapses	32-bit FLP	56.3 mW	Two-level sharing mechanism	XC7Z020 200 MHz

Table 2. Hardware overhead for the performance monitoring platform

Logic	Available	Used	Utilization(%)
Slice LUTs	53,200	1,550	2.91%
Slice registers	106,400	1,109	1.04%

larger number of neurons and synapses while keeping power consumption relatively low (∼56.3 mW), which is an improvement over the approaches of [1, 2] under similar conditions. The current high-performance FPGA is much larger than the one used in this work; therefore, using the proposed approach can accommodate more neurons and synapses.

For the proposed monitoring platform, the hardware overhead is an important metric of platform performance due to that the FPGA device hardware resources are limited. In this platform, only a signal collection module is needed in the FPGA device. The hardware resources used by the monitoring logic is shown in Table 2, which include 1,550 Slice LUTs and 1,109 Slice registers, respectively, i.e. only 2.91% and 1.04% of the resources of the FPGA devices in the Xilinx Zynq-7000 device.

4 Conclusions

This paper proposes a compact layer-tile architecture for the neurons, and an efficient routing architecture for the interconnections between the neurons. The proposed architecture offers an efficient solution to address the problems of area demanding and dense interconnected requirements of the hardware SNNs. It has the capability of scaling to large scale hardware multilayer SNNs. In addition, the performance monitoring mechanism provides an auxiliary functional verification and performance analysis for the SNNs.

Acknowledgments. This research was supported by the National Natural Science Foundation of China under grants 61603104 and 61661008, the Guangxi Natural Science Foundation under grants 2015GXNSFBA139256 and 2016GXNSFCA380017, the funding of Overseas 100 Talents Program of Guangxi Higher Education, the Research Project of Guangxi University of China under grant KY2016YB059, Guangxi Key Lab of Multi-source Information Mining & Security under grant MIMS15-07, the Doctoral Research Foundation of Guangxi Normal University, the Research Project of Guangxi Centre of Humanities & Social Sciences - Ecological Environment Forecast and Harnessing in Ecologically Vulnerable Region of Pearl River and Xijiang Economic Zone (ZX2016030), and the Innovation Project of Guangxi Graduate Education (YCSZ2016034).

References

1. Moctezuma, J.C., McGeehan, J.P., Nunez-Yanez, J.L.: Biologically compatible neural networks with reconfigurable hardware. Microprocessors and Microsystems 39(8), 693–703 (2015)
2. Beuler, M., Tchaptchet, A., Bonath, W., Postnova, S., Braun, H.A.: Real-time simulations of synchronization in a conductance-based neuronal network with a digital FPGA hardware-core. Artificial Neural Networks and Machine Learning 7552(6), 97–104 (2012)
3. Pande, S., Morgan, F., Cawley, S., Bruintjes, T., Smit, G., McGinley, B., Carrillo, S., Harkin, J., McDaid, L.: Modular neural tile architecture for compact embedded hardware spiking neural network. Neural Processing Letters 38(2), 131–153 (2013)
4. Carrillo, S., Harkin, J., Mcdaid, L., Pande, S., Cawley, S., Mcginley, B., Morgan, F.: Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive Network-on-Chip routers. Neural Networks 33(9), 42–57 (2012)
5. Schemmel, J., Fieres, J., Meier, K.: Wafer-scale integration of analog neural networks. In: Proceedings of the International Joint Conference on Neural Networks. pp. 431–438 (2008)
6. Carrillo, S., Harkin, J., McDaid, L.J., Morgan, F., Pande, S., Cawley, S., McGinley, B.: Scalable hierarchical Network-on-Chip architecture for spiking neural network hardware implementations. IEEE Transactions on Parallel and Distributed Systems 24(12), 2451–2461 (2013)
7. Maguire, L.P., McGinnity, T.M., Glackin, B., Ghani, A., Belatreche, A., Harkin, J.: Challenges for large-scale implementations of spiking neural networks on FPGAs. Neurocomputing 71(1-3), 13–29 (2007)
8. Iakymchuk, T., Rosado, A., Frances, J.V., Batallre, M.: Fast spiking neural network architecture for low-cost FPGA devices. In: 7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip. pp. 1245–1248 (2012)
9. Tsodyks, M., Pawelzik, K., Markram, H.: Neural networks with dynamic synapses. Neural computation 10(4), 821–835 (1998)
10. Stein, R.: A theoretical analysis of neuronal variability. Biophysical Journal 5(2), 173–194 (1965)
11. Wade, J.J., McDaid, L.J., Santos, J.A., Sayers, H.M.: SWAT: A spiking neural network training algorithm for classification problems. IEEE Transactions on Neural Networks 21(11), 1817–1830 (2010)
12. Fidjeland, A.K., Shanahan, M.P.: Accelerated simulation of spiking neural networks using GPUs. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2010)