Simulation Modeling Of Artificial Neural Networks*

Konstantin Kormilitsyn[1][0000−0002−3305−6604], Pavel Kustarev[2][0000−0001−9326−0837], and Elizaveta Kormilitsyna[3][0000−0002−5182−6940]

[1] ITMO University, Saint Petersburg, Russia
kormilicinkostia@gmail.com
[2] ITMO University, Saint Petersburg, Russia
kustarev@yandex.ru
[3] ITMO University, Saint Petersburg, Russia
sholohova.elizaveta@mail.ru

Abstract. From year to year information processing algorithms based on neural networks(NN) are gaining more and more popularity. Such networks are characterized by a large number of hidden layers and huge amounts of training data, requiring a specialized high-performance device. In the past few years the main platforms for implementing hardware neural networks have been FPGAs and GPUs. In all power-limited scenarios FPGAs are the natural choice. Developers of specialized computing systems are interested in porting these algorithms to embedded computing platforms. However, there are limitations that prevent the use these algorithms for managing technical systems and objects. One of the limitations is the lack of a mathematical apparatus that would formally evaluate the compliance of real-time constraints for artificial neural networks(ANN). To solve this global problem, a particular problem is solved in this paper. There was proposed and tested a method for validating the temporal characteristics of hardware artificial neural networks implemented on FPGA. This method is suitable for use on neural networks transferred to a hardware platform with pre-selected coefficients, and for neural networks trained directly on the target platform. The method is based on the apparatus of queuing networks (QN), which allow the validation of the temporal characteristics of artificial neural networks according with the hardware features of the target computing platform. The presented method has an applied focus. This paper presents the results of experimental testing of the proposed method and proved the feasibility of using the queuing network apparatus for validating the temporal characteristics of artificial neural networks.

Keywords: Artificial neural networks · simulation modeling · queueing networks · FPGA

* Supported by ITMO University Grant №619296

1 Introduction

Artificial neural networks applications are becoming more popular. They are using for control of technical and cyberphysical systems. High performance and energy efficiency are very important for such systems. It can be achieved due to hardware implementation of calculators. In recent years FPGAs and GPUs are main platforms for hardware neural networks. Not only functional predictability of the behavior, but also temporal predictability are critical condition of using neural networks in control systems. Both of these aspects of predictability are not resolved issues. The article is focused on the second aspect - temporary predictability of the behavior. These days there are no established methodologies for the confirmation of real-time requirements implementation. The article discusses the opportunities for modeling of hardware neural networks which are based on FPGAs using queueing networks.The work shows that [1] different structures of hardware neural networks are well transferred to queueing networks models. Moreover, [2] analysis of functional dynamic parameters with help of mathematical apparatus and queueing networks simulation tools could optimize the architecture of hardware neural networks to achieve real-time limitations.

2 Subject area overview

In this paper it is proposed to consider consider a way of neural networks modeling in order to confirm real-time requirements. In the article [3] authors offered a method for estimating delays in neural networks using the Lyapunov method. In that article, there was derived the Lyapunov equation for recurrent neural networks. Authors did not prove that this formula will be true for other types of neural networks. Therefore, the use of this method for modeling various types of neural networks is impractical. These days there are developments in the development of analog neural networks and their subsequent modeling using hardware components. An example of building hardware neural networks based on memristors is presented in article[4]. Furthermore, there is a the concept of modeling these memristors. Using the techniques for developing neural networks that are described in article[5], it is possible to evaluate the temporal characteristics of a neural network constructed using operational amplifiers. Methods for modeling neural networks based on their hardware implementation are presented in articles [4] and [5]. It is not clear from the articles whether it is possible to transform other hardware implementations to the described models. Studies using simulation modeling are also being conducted in this area. Authors in article [6] are offered description of a neuron using pro-networks. Applying the described methodology, it is possible to simulate the temporal characteristics not only for an individual neuron, but also for the network as a whole. The described method allows to simulate hardware neural networks transferred one in one to the hardware platform. The authors of these articles did not consider cases of insufficient resources of a computing platform. Therefore, it is not clear whether it is possible to simulate optimized neural networks using this method. In the

article [7] correlation between neural network models and queuing network is described. Using the developed model, the authors studied the stability of the neural network to the effects of external signals. Unfortunately, this model was not used for real-time networks. Summing up, the problem of modeling real-time hardware neural networks has not yet been solved and research in this area is relevant.

3 Research

Artificial neuron consists of multi input adder, unit of activation and set of coefficients (synaptic weights, offset, activation function parameters) that are stored in memory. The memory is also used for hardware tabled realization of function‘s activation. Among modern technologies, only FPGA is able to provide a lot of independent memory units with superfast access. Thus, it caused the popularity of FPGAs for ANN development and this also means that the memory can be the bottleneck during realization ANN based on FPGAs.

The structure of NN can be portable on structure of open queueing network (OQN). Each neuron is a node of OQN, which can be presented as a service device without queue. The time of service is determined by speed of FPGA. Despite of visible ”hard” digital circuit synchronization, the time of service is not fixed and has Gaussian distribution. It is explained by the jitter of the clock signal which is accumulated while sequential calculation of ANN layers. Input data stream (requests) is random, evenly distributed. As mentioned, internal memory is the bottleneck in ANN development based on FPGAs. To save memory it is possible to use one unit of function‘s activation by several neurons. Unfortunately, because of parallel calls of neurons to memory units there are collisions and delays. Moreover, there is an issue of NN topologies optimization using performance criterion. The paper describes the developed OQN models, performed simulation, calculated characteristics for ANN-FPGAs with different ways of neuron combinations. The main ways to combine neurons are combine inside layers and between layers. These options are presented on Fig. 1, each marked as A, B, C. OQN devices numbers are marked by digits. These devices perform the neurons functions.

As properties of OQN are determined by properties of network nodes, it is necessary to calculate characteristics of each node individually. Nodes characteristics calculation is made considering temporal characteristics for tabled FPGA realization of neuron with average working time 90 microseconds. These characteristics are presented in Table 1.

Simulation proved dependence of temporal characteristics on node-neuron combination (Table 1 and 2). Using simulation tools it was revealed that some requests in node 2 stayed in a queue, thereby node 2 has become the bottleneck while increasing flow request. It is impractical to have a large amount of memory for other nodes in case of described request flow. Given earlier QN model option (FPGA-ANN) showed the potential evaluation of temporal network characteristics. Futhermore, due to simulation tools this option defined some systems

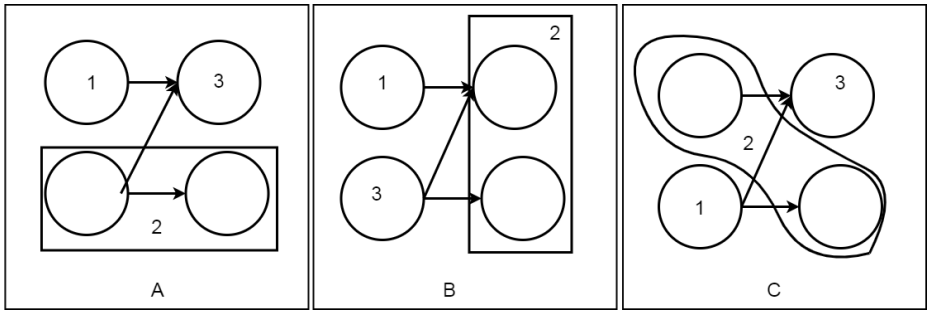

Fig. 1. Options of ANN-FPGA neuron combinations

Table 1. OQN nodes characteristics

Nodal characteristics	Formula	Network topology								
		A			B			C		
		1	2	3	1	2	3	1	2	3
Calling rate	$y = \lambda b$	0.443	0.986	0.442	0.442	0.984	0.443	0.443	0.99	0.442
Load	$\rho = 1 - \rho_0$	0.443	0.986	0.442	0.443	0.984	0.443	0.443	0.99	0.442
Down time	$\eta = 1 - \rho$	0.557	0.014	0.558	0.557	0.016	0.557	0.557	0,01	0,558
Idle time	$w = \frac{\rho b}{1-\rho}$	0	0.515	0	0	0.5	0	0	0,6	0
Residence time	$u = w + b$	90.014	199.934	89.981	90.014	199.932	90.014	90.014	199.942	89.981

bottlenecks and pointed how memory resource optimization important is.Thanks to OQN network-wide characteristics there ia s possibility to identify the maximum allowable intensity ot the requests input stream, thereby test the technical parameters of the developed system.

Table 2. Network characteristics

Nodes characteristics	Formula	Network topology		
		A	B	C
Network idle time	$W = \sum_{j=1}^{n} a_j w_j$	1.187	1.023	1.417
Network residence time	$U = \sum_{j=1}^{n} a_j u_j$	381.929	380.917	383.867
The number of expectation requests	$L = \sum_{j=1}^{n} l_j$	0.97	0.94	0.99
The number of requests in the network	$M = \sum_{j=1}^{n} m_j$	2.81	2.83	2.87

4 Training of neural networks

Often there is a necessity to provide the training of neural networks directly on target computing platform. It is possible to determine three main ways how to provide the training of neural networks:

- training with a teacher - The training with a teacher assumes the availability of the complete set of labeled data for model training at all development stages. The availability of complete marked dataset means that every example in training set must comply with the answer that the algorithm should receive. Generally the training with a teacher is used to solve classification and regression problems. In classification problems the algorithm offers discrete values that correspond to class numbers to which the objects belong. Regression problems are related to continuous data.
- training without a teacher - In the training without a teacher the model has the data set and there is no exact directions of what to do with this data. Neural network itself tries to find data correlations by extracting useful features and analyzing them. Typically, such training method is used by neural networks the purpose of which is to group data according to certain parameters.
- training with a partial involvement of a teacher - Training data set contains labeled data as well as unlabeled.
- training with a support — It is one of machine learning methods during which the test system (agent) is trained, interacting with some environment.

In this paper the method of training with a teacher will be considered. Usually the backpropagation method is used for neural networks training. The main purpose of this method is propagation of error sygnals from the network output to its input. The propagation occures in the opposite direction to the direct propagation of signals in normal operation. Therefore, to realize this method on hardware computing platform is necessary to add the error computing unit to developed neural network. There are three main ways to error calculation: Mean Squared Error (MSE), Root MSE and Arctan. These days there is no unified algorithm for error calculation, so the neural network developer should choose himself the method that is the most suitable for target goal. To calculate the error using method Arctan is necessary to use the following formula:

$$ArtanError = \frac{arctan^2(i_1 - a_1) + arctan^2(i_2 - a_2) + ... + arctan^2(i_n - a_n)}{n}$$

$$= \sum_{i=1}^{n} (\frac{arctan^2(i_i - a_i)}{n})$$

where:

- i - predictions (expected values or unknown results),
- a - observed values (known results).

The error always will be greater when using the method Arctan. This is due to the method working principle: the larger the difference, the greater the error. To calculate the error using method Root MSE is necessary to use the following formula:

$$RMseError = \sqrt{\frac{(i_1 - a_1)^2 + (i_2 - a_2)^2 + ... + (i_n - a_n)^2}{n}} = \sum_{i=1}^{n}((\frac{(i_i - a_i)^2}{n})^{\frac{1}{2}})$$

The Root MSE method has smallest error. To calculate the error using method MSE is necessary to use the following formula:

$$MseError = \frac{(i_1 - a_1)^2 + (i_2 - a_2)^2 + ... + (i_n - a_n)^2}{n} = \sum_{i=1}^{n}(\frac{(i_i - a_i)^2}{n})$$

When using MSE method the error value will be average compared to Arctan and Root MSE methods. Therefore, the MSE method is used more often. It provides balance in error calculation. The MSE method will be will be considered for error computation in this paper. Structural scheme of error computation using MSE is presented in the Fig. 2.

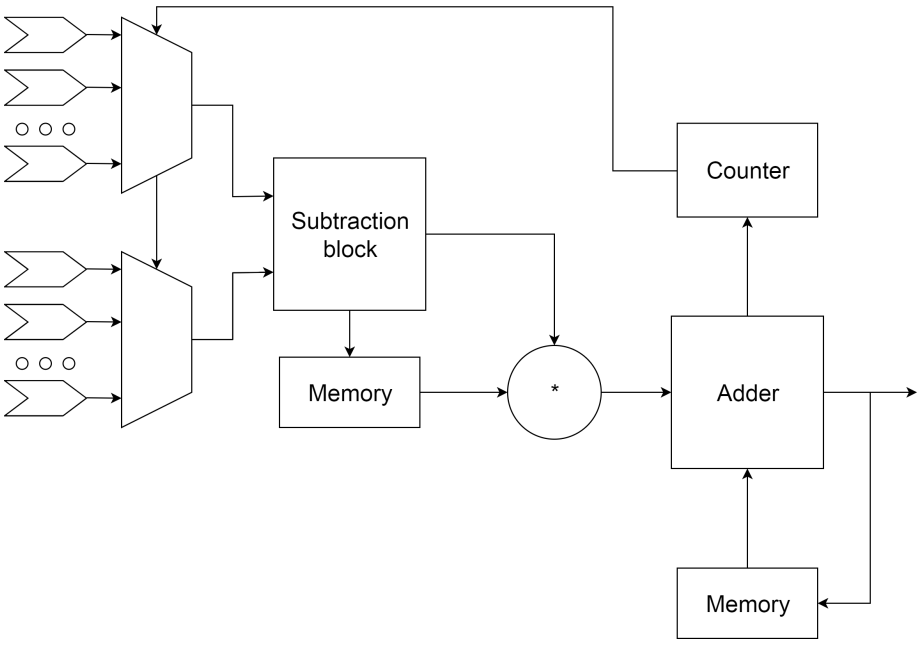

Fig. 2. Structural scheme of error computation using MSE

This structural scheme consists of different units. Firstly, it consists of 2 multiplexers that choose input value for predicted signal and observed signal.

Multiplexer control counter is also included in this scheme. Moreover, the scheme consists of subtraction unit that provides subtracting the observed signal value from the programmable signal value. The difference squaring unit is included in this scheme. It contains the memory unit and multiplier. The scheme consists of adder that provides addition of the value obtained at the output of the squaring unit with the previous value. The output value of the adder goes to the input of memory unit which is reset at the end of the error calculation.

In oder to implement the backpropagation method, it is necessary to calculate the error for each neuron separately starting from the output neuron. The following formula is provided for output neuron error calculation:

$$\delta_o = (i - a) * f^{'}(in)$$

- i - predictions (expected values or unknown results),
- a - observed values (known results),
- $f^{'}(in)$ - derivative activation function

To implement the backpropagation method, it is possible to use only those activation functions that can be differentiate. Derivative function can be represented as follows:

$$f^{'}(in) = f_{sugmoid} = (1 - a) * a$$

This function is provided the neuron error calculation for the inner layer:

$$\delta_h = \sum_{(i=1)}^{n} (w_i * \delta_i) * f^{'}(in)$$

where

- w - the weight of the output value,
- d - the error value of the neighboring neuron.

To calculate the new value of the neuron weight, the following formula is presented:

$$\triangle w_i = E * GRAD_w + a * \triangle w_{i-1}$$

where

- E - training speed,
- a - moment - training step size.

To calculate the new value of the neuron weight, it is necessary to obtain the gradient value for the given function. The following formula is provided the gradient value calculation:

$$GRAD = \delta_b * out_a$$

For artificial neural networks (ANN) shown in Fig. 1, the simulation modeling of the training process was performed. To perform the simulation modeling,

developed model should be supplemented by a device that imitates the error calculation process. Moreover, functions that imitate the neuron work should be replaced by functions that calculate the new neuron weight. Calculation of the nodal characteristics should be performed according to time characteristics for target platform. Average working time is 40 microseconds and equals function of new neuron value calculation. Average time of ANN error calculation is 5 microseconds. All these characteristics are presented in Table 5.

Table 3. OQN nodal characteristics for training process

Nodal characteristics	Formula	Network topology								
		A			B			C		
		1	2	3	1	2	3	1	2	3
Calling rate	$y = \lambda b$	0.192	0.41	0.191	0.191	0.443	0.193	0.192	0.401	0.192
Load	$\rho = 1 - \rho_0$	0.192	0.41	0.191	0.191	0.443	0.193	0.192	0.401	0.192
Down time	$\eta = 1 - \rho$	0.808	0.59	0.809	0.809	0.557	0.807	0.808	0.599	0.808
Idle time	$w = \frac{\rho b}{1-\rho}$	0	0.214	0	0	0.225	0	0	0.201	0
Residence time	$u = w + b$	41.117	87.109	40.915	40.898	88.754	40.915	41.106	86.931	40.915

Dependence of time characteristics on the option of combining neuron nodes was also confirmed for the neural networks simulation training process (Table 5 and Table 4). The simulation modeling results show that in contrast to the process of calculating the time characteristics of the output signal the best way to combine neurons for training process is presented in Fig. 1.C. This is due to the fact that in this case there is a multiplexing of two unconnected neurons, thus the backpropagation method is calculated without delay.

Table 4. Network characteristics for the training process

Nodal characteristics	Formula	Network topology		
		A	B	C
Network idle time	$W = \sum_{j=1}^{n} a_j w_j$	0.516	0.616	0.44
Network residence time	$U = \sum_{j=1}^{n} a_j u_j$	169.141	170.55	168.952
The number of expectation requests	$L = \sum_{j=1}^{n} l_j$	0.42	0.43	0.39
The number of requests in the network	$M = \sum_{j=1}^{n} m_j$	1.27	1.31	1.28

5 Experiment

The correctness of simulation results with using OQN was checked full-scale experiment. The stand consists of:

- ANN realizes on FPGA by Lattice MachXo2-1200ZE.
- Input-output data system for neural networks. Input data are pre-buffered as FIFO and synchronously extracted to ANN inputs. There is a requests stream with a certain, adjustable intensity. The data trigger on the network output allows to measure the delay in data processing relative to the input clock signal.
- The monitor which configures input FIFO while system start. It checks validity (availability) of the output data and determines the temporal NN characteristics.

The experiment consisted of 1000 sequentially applied input impacts with fixing the input signal installation time and observable system's response time. ANN average time reaction data are presented in Table 5, and histogram of the distribution density of the ANN reaction for network topologies type A is presented in Fig. 3.A.

Table 5. Network characteristics

Characteristics	Formula	Network topology		
		A	B	C
Reaction time	$u = \frac{\sum_{i=0}^{n} u_i}{n}$	384.763	377.178	390.375

As presented in Table 5, ANN reaction time, which is obtained in experiment, coincides (up to 5%) with average time of staying request in the network, calculated for OQN model. Calculation error is related to the fact that time limitations of hardware interfaces between individual network neurons were not taken into account in the OQN model. Thus, ANN simulation results based on OQN were confirmed. Furthermore, because of the information transfer between individual neurons it is necessary to include in the OQN model the delay. This delay can be implemented as separate service device which simulates operation of a separate communication interface. As a result changes in data transfer interfaces between individual neurons can lead to changes in the OQN model. Identified instability of calculated reaction time is due to the fact that functioning of ANN hardware unit in FPGA is not absolutely synchronous. As the tacts number for ANN output calculation is constant in this realization, instability as caused by clock signal jitter. It is assumed that the factor causing the ANN clock signal jitter is the primary source of clock pulses. Generator signal frequency measurements and distribution histogram presented in Fig. 3.B were made. The nature of the frequency values distribution of the clock generator (Fig. 3.B) coincides with

the nature of the ANN reaction time distribution (Fig. 3.A), that allows to fix direct connection of the second fact with the first. Therefore, developing ANNs based on FPGA, special attention should be paid to the stability parameters of the used clock signal generator.

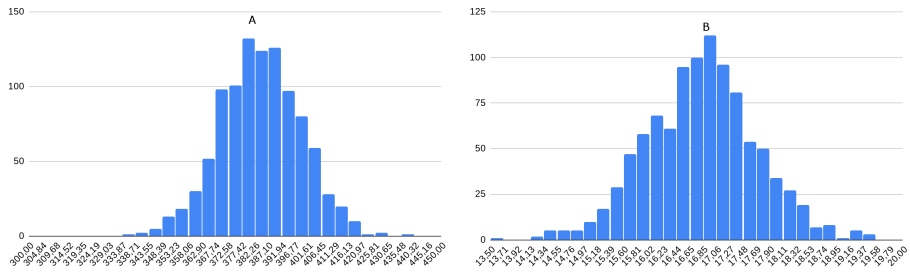

Fig. 3. Distribution density

Previously described experiment was also performed for neuron networks training process. The Table 6 shows that ANN training time coincides with the calculated values obtained in the OQN model simulation. Coincidence accuracy is 5%. Network topology type C provides the best reaction time result for simulation modeling too. Reaction density distribution graph that was created for OQN training process is presented in Fig. 4. It revealed that the nature of the distribution of the frequency values of the clock generator (Fig. 3.B) coincides with the nature of the distribution of the OQN reaction time (Fig. 4).

Table 6. Network characteristics for training process

Characteristics	Formula	Network topology		
		A	B	C
Reaction time	$u = \frac{\sum_{i=0}^{n} u_i}{n}$	169.074	169.891	168.06

6 Conclusion

The study concluded that applying OQN models for hardware neural networks simulation based on FPGA is expedient and using this mathematical apparatus it is possible to evaluate the temporal characteristics of hardware NN. The results of this method were confirmed by field tests. It was revealed that one of the source of unstable operation of hardware NN is associated with a jitter synchrosignal. The correlation of the distribution characteristics of the frequency

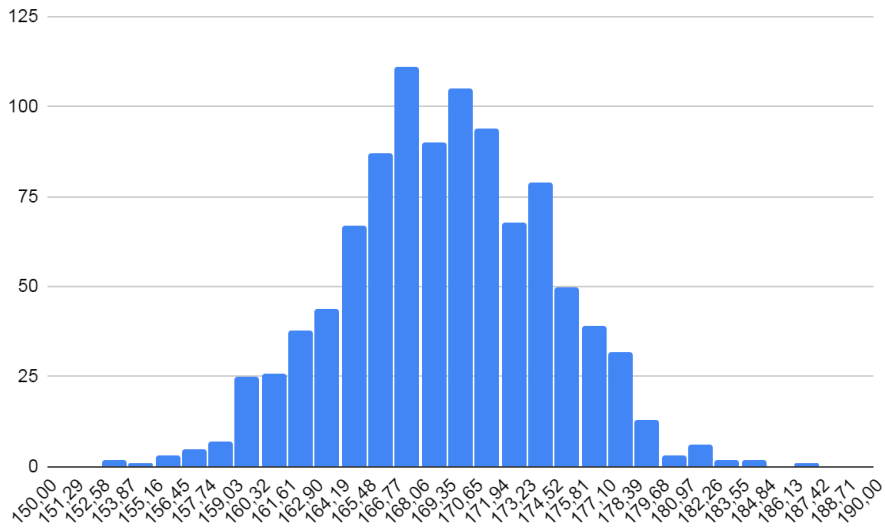

Fig. 4. Distribution density for training process

values of the clock generator with the distribution characteristics of the ANN reaction time was also proved experimentally. This experiment was carried out on ANN implemented on a single crystal and with constant environmental characteristics. In further studies it is planned to test this theory for multi-chip ANNs implementation.

References

1. Himavathi, S., Anitha, D. and Muthuramalingam, A.,Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization, IEEE Trans. on Neural Networks, 18(3), 880-888(2007)
2. Turkovsky Y.A., Bogatikov E.V., Tikhomirov S.G., Adamenko A.A. Modeling the restoration of biological and biotechnological systems using hardware analog and software artificial neural networks - Bulletin of the Voronezh State University of Engineering Technologies
3. Jinde Cao ,Jun Wang Global Exponential Stability and Periodicity of Recurrent Neural Networks With Time Delays , Senior Member, IEEE
4. D.D. Kozhevnikov, N.V. Krasilich. Memristor-based Hardware Neural Networks Modelling Review and Framework Concept, Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp. 243-258. DOI: 10.15514/ISPRAS-2016-28(2)-16
5. Novotarsky M.A., Simulation of neural networks for solving equations of mathematical physics by local-asynchronous methods ,Radioelectronics. Computer science. Management., No. 1. , 2001
6. A. Muthuramalingam, S. Himavathi, E. Srinivasan, Neural Network Implementation Using FPGA: Issues and Application , International Journal of Information Technology, vol. 4 Number 2, 2008 pp. 86–92.

7. Erol Gelenbe. G-networks, A unifying model for neural and queueing networks ,Annals of Operations Research, vol. 48, issue 5, 1994, pp. 433–461.