Optimization of Hardware Neural Networks using Queuing Theory

Konstantin Kormilitsyn[1], Pavel Kustarev[1] and Tatyana Malysheva[1]

[1]ITMO University,Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia

Abstract

Neural networks are gaining more and more popularity today. The scope of their application is quite extensive and covers segments from banking to medical research. Neural networks are currently the most popular application models. These networks, characterized by a large number of hidden layers and huge volumes of training data, require specialized high-performance devices. FPGAs and GPUs have been the main platforms for implementing hardware neural networks in the past few years. A critical condition for the use of neural networks in embedded systems is predictability in time.At the same time, there are a number of limitations that prevent the use of neural networks in portable embedded systems. There are two main requirements for such systems: power consumption and real-time requirement. In all power-constrained scenarios, FPGAs are the natural choice. Currently, there are no well-established methodologies for confirming compliance with real-time requirements. The author investigates the ability to simulate FPGA-based hardware neural networks using queuing networks (QNN). The article shows that various structures of hardware neural networks are well mapped to the QNN model and the analysis of functional and dynamic parameters using the mathematical apparatus and simulation tools of QNN will allow optimize the architecture of hardware neural networks to achieve real-time constraints. In addition to optimizing time parameters, this article discusses the use of QNNto optimize memory resources.

Keywords

Artificial neural networks, simulation modeling, queueing networks, FPGA

1. Introduction

Nowadays neutral networks are the most popular models of applications. These networks, which are characterized by a large number of hidden layers and huge amount of data for training, need specialized high performance apparatus. In the last few years, FPGA and graphics processors are the main platforms for implementation of hardware neural networks. Due to limitations, it is impossible to use neural networks in portable embedded systems. There are two main requirements for such systems: energy supply and real time requirement. FPGA is natural choice in all cases with limited power consumption. Considering the amount of layers and neurons in networks, which tend to infinity, required memory resources also tend to infinity. Thus developer will run into the lack of resources problem. Moreover, these days there is no

Proceedings of the 12th Majorov International Conference on Software Engineering and Computer Systems, December 10–11, 2020, Online & Saint Petersburg, Russia

kormilicinkostia@gmail.com (K. Kormilitsyn); kustarev@itmo.ru (P. Kustarev); tamalysheva@itmo.ru (T. Malysheva)

0000-0002-3305-6604 (K. Kormilitsyn); 0000-0001-9326-0837 (P. Kustarev); 0000-0002-1171-379X (T. Malysheva)

mechanism of time characteristics description for neural networks, which consider hardware features of equipment. In this way, execution of real time requirements is not guaranteed for neutral networks that is implemented on some platform.

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. 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. Hardware realization of neural network

When hardware implementation of artificial neural network (hardware neural network), the first stage is modeling of artificial neural network. As an example the model which has 4 neurons and 3 layers and also implemented on FPGA will be considered (shown in the Fig. 1). In this implementation memory blocks, multipliers, adder and hardware activation function are used. The second stage of hardware neural networks building is modeling of their hardware implementation, which is necessary also for evaluation of system performance. It should be considered that realized on FPGA neural networks are resource intensive and take large amount of memory on chip [1]. When implementing the neutral network the most expensive in terms of memory is the realization of neurons activation function. There is an opportunity to realize so big neural network that there will not be enough chip memory for this network. Therefore, there is an assumption that current realization of activation function takes 30% of FPGA memory resources. In this case it is necessary to combine two and more neurons in one activation function. The simplest way how to optimize the memory is reuse one activation function for several neurons [2]. In this article the way of combination of two neurons of the second layer is described. Using the following method [7] of network design, it is possible to implement the neutral network as it shown on Fig. 1. The number of neurons and multipliers is reduced in this realization. The units of multiplexer, demultiplexer, memory and resources manager are added. Last unit is responsible for choice of current neuron, memory unit stores

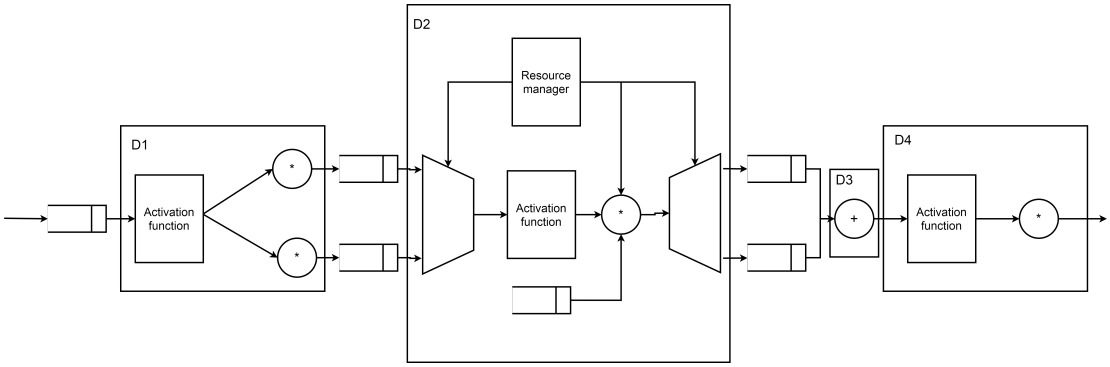


Figure 1: Neural network realization on FPGA.

coefficients for each neurons. The choice of neurons combination imposes restrictions on real time requirements execution. Considering a huge variety of options of neurons combination it is impossible to solve the problem using exhaustive search with field tests. Moreover, it is impossible to prove real time requirements execution conducting only field tests. Thus, it is need to model the system in view of resource optimization model which provides hardware platform needs. As a result, to date, there is unresolved issue of analysis of real time characteristics for neural network hardware implementation. Neural network structure can be displayed on open queueing network structure (OQN). Each neuron is OQN node and can be represented as service devices without queue. Neuron service time depends on FPGA performance. Despite hard synchronization of digital circuit, service time is not fixed and has Gaussian distribution. This is due to clock jitter which accumulates during calculation of artificial neural network (ANN) layers. Network input data (tasks) is random and uniformly distributed. As already said, FPGA implementation bottleneck is internal memory. Using of one unit of activation function is the way to save memory. Unfortunately, because of neurons parallel calls to memory units there are collisions and delays. Also using of one unit of activation function sets a task of neural network topology optimization by performance criteria. The Fig. 1 shows special case of combination of two connected neurons in one. This model is applicable in cases of combination of two and more neurons from one layer, where each output is connected to the same neurons from the next layer. In this article probability of tasks loss during the transaction from one unit to another is not considered, because the network is implemented on one chip on which the probability of sygnal loss during the transaction tends to zero. Since properties of open queueing network are largely determined by properties of network nodes, it is necessary to calculate characteristics of each node separately. Calculation of nodes characteristics will be execute considering neurons time characteristics which are described in the following article [8]. Calculated parameters are presented for described (D) queueing network and for optimized (O) (optimized queueing network will be reviewed in paragraph «Hardware neural network optimization») and shown in Table 1.

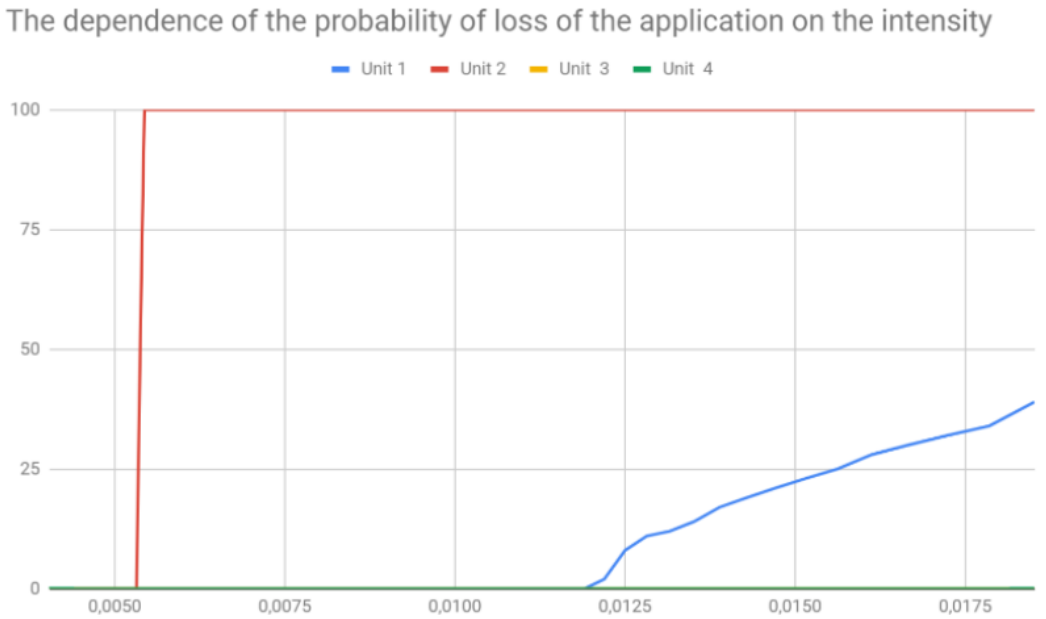
As shown in Table, the node 1 and the node 4 are approximately equally loaded. As expected, the node 2 has the highest load. This is due to the fact that the task sequentially is processed in two phases on the node 2. Using simulation modeling there was a conclusion that only

some tasks remain in the queue on the node 2, thereby the node 2 will become the bottleneck with increasing flow rate. To have a large amount of memory is impractical for other nodes with described tasks stream. To prove this statement on practice, it is necessary to conduct the system simulation with different intensity for input data stream thereby identifying maximum network throughput. The Fig. 2a shows a plot of the flow rate versus the time the task was in the each node. The graph is observed the intermittent change of task service time on the node 2. To explain this system behavior was built a plot of the probability of the task loss versus the intensity which is presented in the Fig. 2b. As can be seen from the graph, the node 2 goes into the failure state first. As with 100% probability of failure tasks do not enter the device for service, it is possible to observe zero service time in the node. Thereby experimentally proven, that the node 2 is the bottleneck of the system. Probability of task loss on the node 1 is increasing after the node 2, as flow rate is higher than performance of the developed node. The flow rate change does not affect the probability of task loss on node 3 and node 4. This is due to the fact that there is a serial node connection in the considered system, consequently intensity of task entrance identified by performance of the node 2, but not by the intensity of the task entrance.

Table 1
Queueing network node characteristics

Nodal characteristics	Network topology							
	Node 1		Node 2		Node 3		Node 4	
	D	O	D	O	D	O	D	O
Calling rate	0.443	0.443	0.984	0.443	0.049	0.049	0.442	0.975
Load	0.557	0.557	0.016	0.557	0.951	0.951	0.558	0.025
Idle time	0	0	0.515	0	0	0	0	0.513
Residence time	90.014	90.014	199.934	90.014	10.002	10.002	89.981	199.931

(A)



(B)

Figure 2: Dependence of queueing network characteristics on stream intensity.

4. Optimization of hardware neural network

The bottleneck must be unloaded dor system optimization. As mentioned earlier, the bottleneck is node 2. Since in this problem there is no way to change the network topology to the bottleneck offload, the only way is to increase the performance of the overloaded node, due to the fact that it is not possible to directly increase the performance of nodes. To change the layout of neurons and the service discipline is the only way to increase productivity. For example, neurons from levels 2 and 3 were combined. The combined neurons formed Node 4. Presented in Table 1 (O) nodal characteristics were calculated for the described model. As can be seen from the table, this action allowed to slightly unload the Node 4.Thus, it is obvious that the performance of a hardware neural network is affected not only by the activation function performance, but also by the way neurons are combined. Various methods for neurons combining are presented in Fig. 3. Also, the discipline of the task service in the device (resource manager) affects performance. Thus, when simulating hardware networks, the model should take into account specific features of the resource manager. Therefore, during the creation of the model, it is necessary to pay attantion not to the existing models of simulation modeling, but to models of hardware resource managers that can be implemented on the target platform. There are three main models for implementing resource managers which are executed on FPGAs: priority processing, non-priority processing and pipeline processing. The combination of two neurons of different levels are shown in the Fig. 3a. In this case, a classic example of tasks race is presented. Since the neuron from third level has two inputs, it is possible that a task for one input arrives earlier than the second. There are two main service models which be distinguished in this example: priority processing and pipeline processing. Priority processing means that tasks arriving at the second level of the neural network receive a higher priority than tasks arriving at the third level. Pipeline processing means that tasks arriving at the second level and third level have the same priority and processed sequentially. Moreover, a waiting phase is added between the first and second phases in addition to the execution phases. Waiting phase is equal to the task critical path from level 2 of the neural network to level 3. To analyze the two models, was build a plot of the dependence of the output stream intensity versus the input stream, the plot is presented in Fig. 2a. As can be seen in the case of priority applications, the output intensity tends to zero. This is due to the fact that with an increasing intensity of the input signal, all the computational time is allocated to the task with high priority, thereby tasks are not received to the second phase and begin to accumulate endlessly in the queue, which can be seen in Fig. 2b (graphs of the dependence of the queue size on the input stream intensity). Thus, for this example, using the sequential processing model, it is possible to reduce the required amount of memory in the queue and increase the reliability of the system. Fig. 3b shows an example of combining neurons of the same level. In this case, flow races may also occur. Unlike case 4a, the priority service model can not be applied to this example, since the nodes have equal priority and cannot be considered in any other way. Therefore, there are two models for task s ervicing: pipeline services and equal priority services. Thus, an incoming task with equal probability will be served on phase 1 or phase 2. Moreover, for analysis were also build graphs of the dependence of the output stream intensity on the input stream and graphs of the dependence of the queue size on the intensity of the input stream. The graphs are presented in Fig. 4a and Fig. 4b respectively.

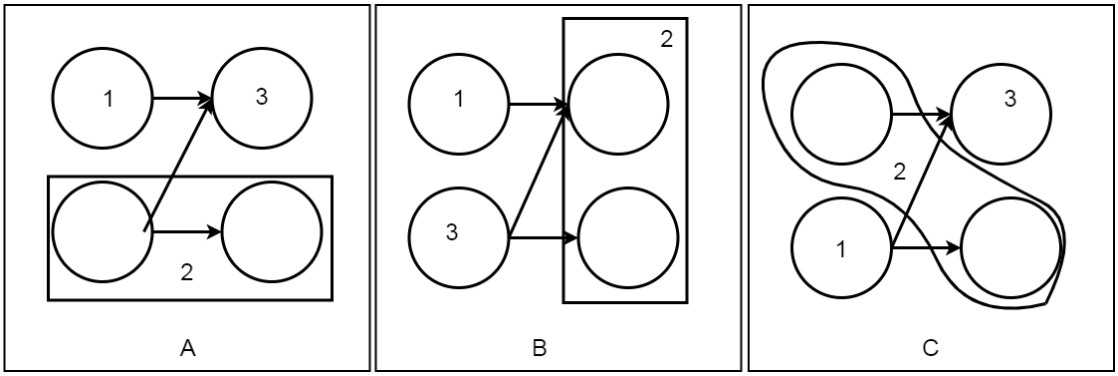
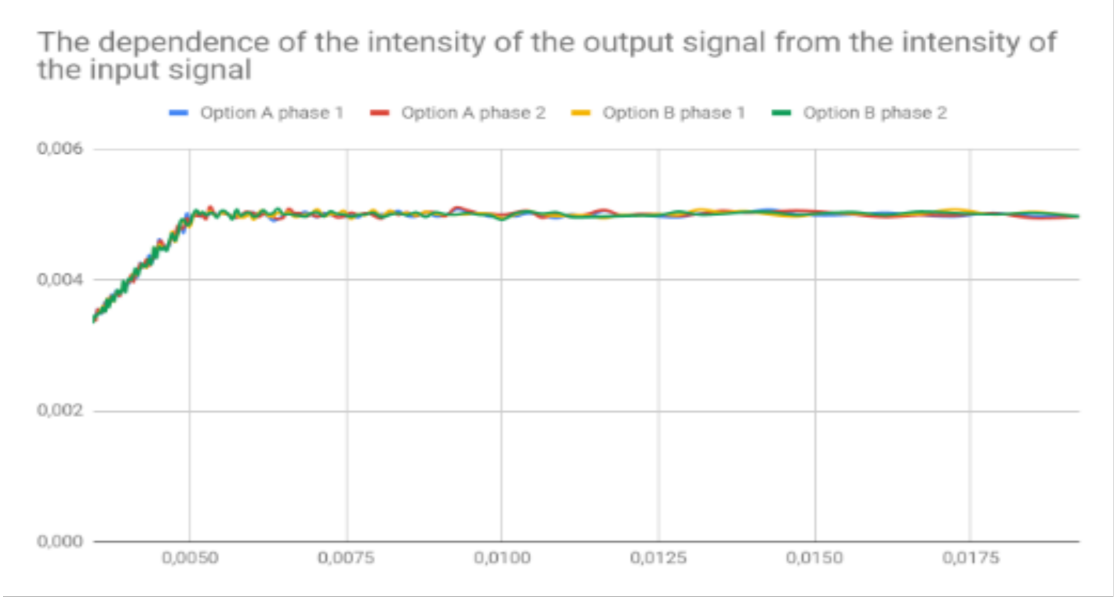


Figure 3: Example of the neurons combination.

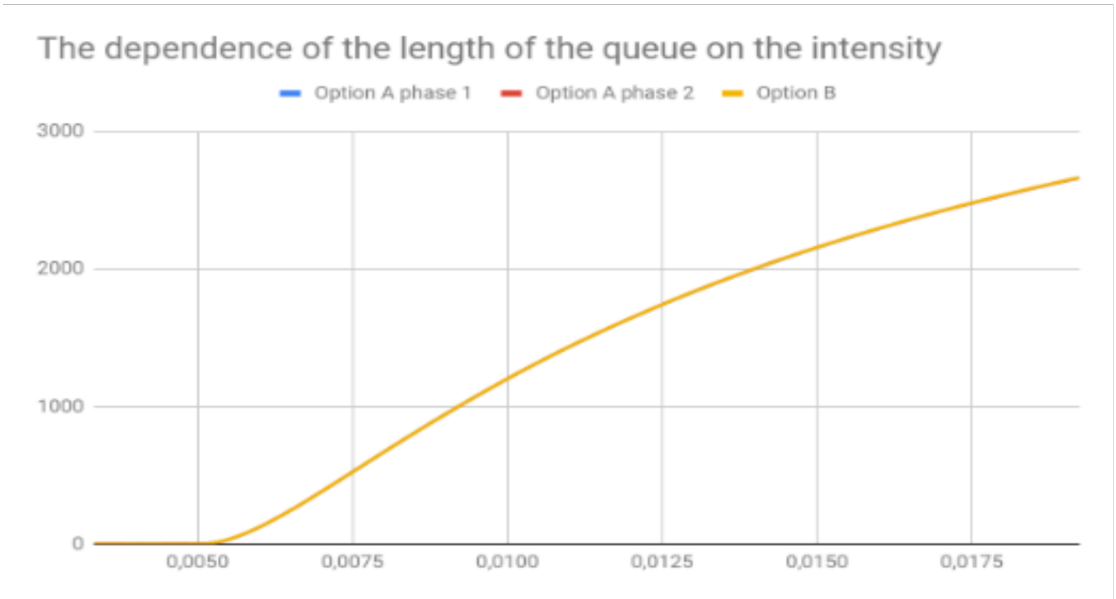
As can be seen from the graphs, these models are equivalent, which means that the choice of service discipline does not have a strong effect on the model as a whole. Fig. 3c shows an example of combining unconnected neurons of different levels. Unconnected neurons are those in which the neuron output does not enter the input of another. When combining this type of neurons, there are possible following cases: neuron combination can be at the distance of one layer or several layers. Thus, a sequential processing model is not applicable in this case, since it will lead to an increase in the task waiting time in the second phase, depending on the distance of the neurons from each other. Priority and non-priority service models were also considered. For analysis, the above graphs were also plotted, presented in Fig. 5a and Fig. 5b respectively. As can be seen from graphs, using priority service discipline there is a possibility of monopolization of computer resources for tasks with higher priority. It means that failures in service of tasks from another task class are possible. Further, the calculation of the nodal characteristics will be executed considering the time characteristics for a tabular implementation of a neuron with an average operating time of 90 s. These characteristics are presented in Table 2.

Table 2
Network characteristics of the open queueing network

Nodes characteristics	Network topology		
	A	B	C
Network idle time	1.187	1.023	1.417
Network residence time	381.929	380.917	383.867
The number of expectation requests	0.97	0.94	0.99
The number of requests in the network	2.81	2.83	2.87

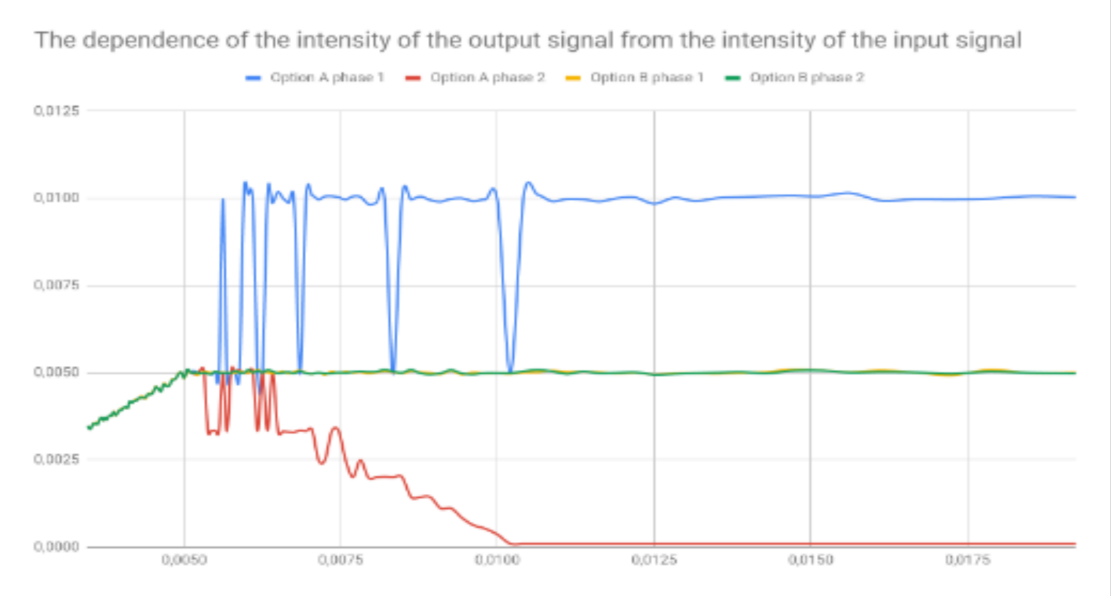


(A)

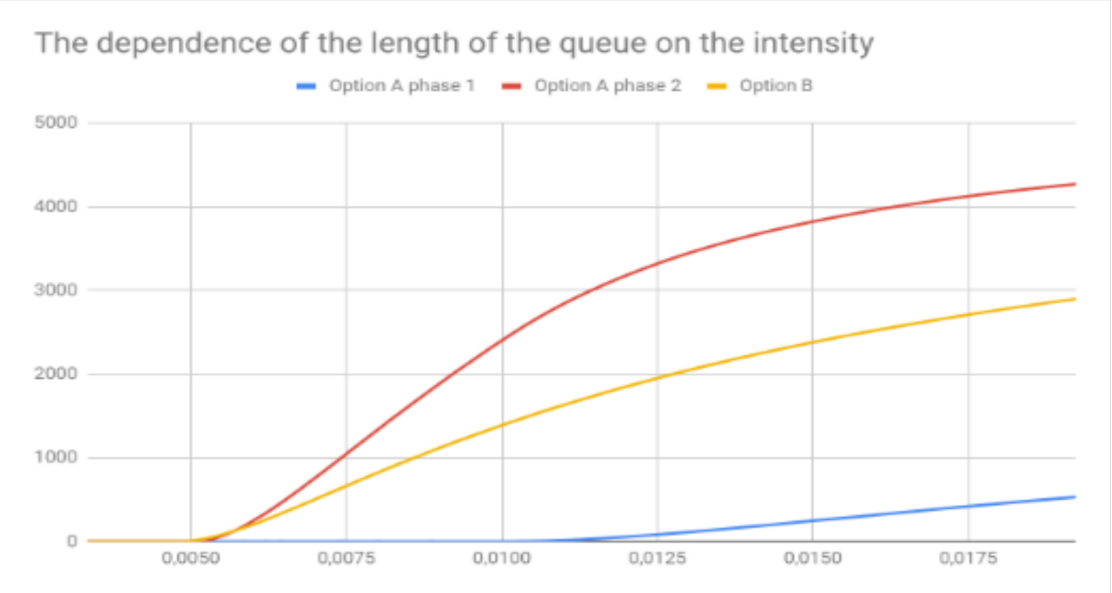


(B)

Figure 4: Dependence of nodes characteristics on stream intensity(a).



(A)



(B)

Figure 5: Dependence of nodes characteristics on stream intensity(b).

5. Conclusions

As a result of the study, it was concluded that the use of the OQN model for modeling hardware neural networks is expedient and using this mathematical apparatus it is possible to estimate

the temporal characteristics of hardware neural networks. The OQN model allows simulating a system with different layouts of neurons and disciplines of their maintenance. Thus, it is possible to choose the optimal network layout in terms of performance and system failure probabilities. OQN models are excellent for describing both parallel and sequential processes that take place in the hardware implementation of neural networks. Consequently, using the OQN mechanism, it is possible to assess the fulfillment of real-time requirements, the load of memory elements and the fault tolerance of the system, as a result, increasing the reliability of hardware implementations of neural networks.

6. Acknowledgments

Supported by ITMO University Grant №620164

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, Vol. 18 ,No. 3,Year 2007, pp. 880-888.

[2] Turkovsky Y.A., Bogatikov E.V., Tikhomirov S.G., Adamenko A.A., “Modeling the restoration of biological and biotechnical systems using hardware analog and software artificial neural networks”, Bulletin of the Voronezh State University of Engineering Technologies,Vol. 12 ,No. 2,Year 2018, pp. 86-92.

[3] Jinde Cao, Jun Wang, “Global Exponential Stability and Periodicity of Recurrent Neural Networks With Time Delays”, IEEE Transactions on Circuits and Systems I: Regular Papers,. Vol. 52,No.5,Year 2005, pp. 920-931

[4] D.D. Kozhevnikov, N.V. Krasilich, “Memristor-based Hardware Neural Networks Modeling Review and Framework Concept”, Proceedings of the Institute for System Programming of the Russian Academy of Sciences, Vol. 28 ,No. 2,Year 2016, pp. 243-258.

[5] Novotarsky M.A., “Simulation of neural networks for solving equations of mathematical physics by local-asynchronous methods”,Radio electronics, informatics control, Vol. 1 ,No. 5,Year 2001, pp. 113-116.

[6] A. Muthuramalingam, S. Himavathi, and E. Srinivasan, “Neural network implementation using FPGA: issues and application,”, International Journal of Information Technology, Vol. 4, No. 2,Year 2008, pp. 86–92.

[7] S.Himavathi; D. Anitha; A. Muthuramalingam Feedforward “Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization”, IEEE Transactions on Neural Networks, Vol: 18, No. 3 ,Year 2007,pp. 880-888.

[8] D.Khomenko I.V., Lepetaev A.N. “Study of the influence of the design parameters of a quartz resonator on the spectrum of natural frequencies”, Dynamics of systems, mechanisms and machines , Vol: 4, No. 2 ,Year 2012,pp. 184-187.