Optimized FPGA Implementation of an Artificial Neural Network Using a Single Neuron

Yassen Gorbounov[1(✉)] and Hao Chen[2]

[1] New Bulgarian University and MGU "St. Ivan Rilsky", Sofia, Bulgaria
ygorbounov@nbu.bg
[2] China University of Mining and Technology, Xuzhou 221000, People's Republic of China
hchen@cumt.edu.cn

Abstract. Since its emergence in the early 1940s as a connectionist approximation of the functioning of neurons in the brain, artificial neural networks have undergone significant development. The trend of increasing complexity is steadily exponential and includes an ever-increasing variety of models. This is due on the one hand to the achievements in microelectronics, and on the other to the growing interest and development of the mathematical apparatus in the field of artificial intelligence. It can be assumed however that overcomplicating the structure of the artificial neural network is no guarantee of success. Following this reasoning, the paper proposes a continuation of the author's previous research to create an optimized neural network designed for use on resource-constrained hardware. The new solution aims to present a design procedure for building neural networks using only a single hardware neuron by using context switching and time multiplexing by the aid of an FPGA device. This would lead to significant reduction in computational requirements and the possibility of creating small but very efficient artificial neural networks.

Keywords: Artificial Neural Network · Contextual Switching · Hardware Acceleration · FPGA · Optimization

1 Motivation

The study of the principles and working patterns of the nervous system of living organisms has a long history. The first attempts to recreate it with technical means date back to the early 1940s, when the neurophysiologist Warren McCulloch, and the mathematician Walter Pitts published a research on possible working model of the neuron [17]. Their point of view has been oriented toward the use of electronic circuits to model simple neural networks. In general, the modeling of artificial neural networks (ANN) aims at not only recreating the basic functionality of the neural cell alone but also mimicking intelligence at the biological level by simulating the neurophysiology of the brain. The ultimate goal is to achieve such a mechanism of information processing, which is as close

T. Zlateva and G. Tuparov (Eds.): CSECS 2023, LNICST 514, pp. 233–246, 2023.
https://doi.org/10.1007/978-3-031-44668-9_19

as possible to the processes taking place in a set of multi-connected neurons, and then make all this work in a hardware computation device. Thus the information processing is parallel and distributed among multiple simple, interconnected elements. In theory, a single nonlinear layer with a very large number of neurons can learn arbitrary relations between the input and the output of that layer. Increasing the number of layers and the ways their individual nodes (neurons) are interconnected, may lead to an even more efficient learning process [16]. ANNs are capable of solving a large variety of linear and nonlinear problems, so in a more general mathematical sense, neural networks are considered by [23] as a universal approximator. The final objective of these models is to create a technical device capable of learning and making decisions similar to the human brain and at the same time outperforming its speed of data processing.

A consistent overview of the major ANN topologies, taxonomy, and chronological development is provided by the members of the Asimov Institute [16]. Without claiming to be exhaustive, in Table 1 are outlined the major milestones in the development of ANN [3, 18, 25, 29].

Table 1. Key milestones in the evolution of the ANN development.

Year	Authors	Achievement
1943	W. McCulloch, W. Pitts	Modeled a simple NN with electronic circuits [17]
1949	D. O'Hebb	The learning hypothesis of biological neurons – Hebbian Learning [9]
1957	F. Rosenblatt	Perceptron, the oldest ANN model still in use today [27]
1959	B. Widrow, M. Hoff	(Multiple) ADAptive LINear Elements – ADALINE, MADALINE [34]
1969	M. Minsky, S. Pappert	Demonstrate the impossibility for a single-layer perceptron to learn an XOR function [19]
1974	P. Werbos	Backpropagation – backward propagation algorithm of errors propagation working back from output nodes to input nodes [33]
1982	J. Hopfield	Hopfield network. A content-addressable model for understanding human memory [10]
1998	LeCun	Convolutional Neural Network (ConvNet, CNN). A class of ANN mostly applied for image analysis. It uses the convolution instead of matrix multiplication [15]
1985–2006	Various scientists	Boltzmann Machine (Ackley), Autoencoder (Rumelhart), Multilayer Perception (Rumelhart,Hinton, Williams), Recurrent Neural Network (Jordan), Restricted Boltzmann Machine (Smolensky), LeNet (LeCun), Long short-term Memory LSTM (Hochreiter, Schmidhuber), Deep Belief network (Hinton)
2005	F. Scarselli, S. Yong, M. Gori, M. Hagenbuchner, A. Tsoi, M. Maggini	Graph Neural Network (GNN). A class of ANN for processing data that are given as graphs [28, 35]
2012	A. Krizhevsky	AleXNET. It is the first fast GPU-implementation of a CNN [14]

(*continued*)

Table 1. (*continued*)

Year	Authors	Achievement
2014	I. Goodfellow	Generative Adversarial Networks (GNN). ML framework with two NN where one agent's gain is another agent's loss [6, 32]
2020	OpenAI	Generative Pre-trained Transformer 3 (GPT-3), a deep learning model to produce human-like text [1]

It can be seen from this table that the diversity and the capabilities of ANNs are constantly rising all along with the increase of their complexity. There are authors however that claim that "fewer neurons are needed as experience is gained". Such an observation has been made by the renowned professor of molecular and cell biology at the University of California, Berkeley, Walter Freeman, who is among the founders of computational neuroscience, where mathematics is employed to study brain dynamics. He concludes that "after sniffing again and again it appears that only a few neurons are sufficient", and "animals and humans can perceive same information for things like odour, but require different amount of neurons – flies have 100 000 neurons while human brain has billions" [4]. McCulloch and Pitts with their cybernetic neuron model [17], and Karl Pribram [24] with his studies on brain dynamics, have strongly impacted Freeman's scientific thought. This interpretation differs from theories in neuroscience, which aim to collect data from as many neurons as possible, which can then be analyzed offline to improve understanding of the brain's information processing [13].

Apart from the main task of solving the function approximation, object recognition, or decision-making problem, ANNs nowadays face the need to overcome the problem of saving power. The latter is obvious, since the number of neurons in the mathematical model, the topological complexity, and therefore the required computation resources, are growing at breakneck speed. In [36] a very important observation is made, namely, the fact that a considerable amount of arithmetic operations in ANN do repeat many times, so in order to decrease the energy consumption redundant computations can be eliminated. The authors of that study called their model CORN (COmputation Reuse-aware Neural network accelerator) and proposed the neurons to share their computation results. Another study [20] discusses a multiplexing technique called DataMUX which allows ANN to analyze multiple data streams in nearly simultaneous manner by multiplexing the inputs and the outputs. In this study, the software approach has been taken as the experimental platform and no hardware implementations have been attempted. A recent paper in Nature Communications [30] discusses the possibility to build a deep neural network using a single neuron. The architecture proposed by the authors is named "folded-in-time DNN (Fit-DNN)" and uses multiple time-delayed feedback loops to model the so called delay dynamical system, again by software means. In fact, to date, the design and implementation of neural networks using high-performance hardware is gaining a lot of speed [21, 22, 26, 33]. However, the trend of increasing complexity, and therefore increasing the cost of computing power, continues to be sustainable.

In the present research, an attempt is made to go even further by allowing the entire artificial neural network to be implemented using a single hardware neuron. For neural

networks with a smaller number of layers and up to a few tens of neurons, this approach can be very efficient and quite sufficient to solve the entire task. For more complex neural networks such as deep neural networks (DNN), convolutional neural networks (CNN), graph neural networks (GNN), etc., this approach can aid in combining a network built using a single hardware neuron and time-multiplexed computations [7] for modeling of one layer or one node in a graph respectively, and next context-switching of the weight matrices and the activation function matrices. The latter can lead to a reduction in the total number of network layers, a significant reduction in hardware resources, and hence strongly decreased energy consumption. This suggests the possibility of creating more affordable, smarter, and energy-efficient devices that employ space-constrained hardware, without making a compromise with the performance.

The remainder of this work is organized as follows: At first in chapter "Hardware model of the artificial neuron," it is described the hardware principle that lies behind the implementation of the artificial neuron. Discussed are its building blocks and the quantization problem. In chapter "The single-layer artificial neural network" the context switching organization proposed in the previous work of the authors [7] is reintroduced as it can aid in understanding the approach presented in the next chapter "The single hardware-neuron network". Directions for future improvement are given in chapter "Future work". The chapter "Conclusions" summarizes the results.

2 Hardware Model of the Artificial Neuron

In living organisms, the nervous system is made up of nerve cells called neurons, which are their basic structural units. An abstract model of the living neural cell together with its analogical mathematical model is depicted in Fig. 1. It puts together the biological terms (A), the corresponding mathematical abstract counterparts (B), and the basic task that is performed in the process of converting the input to output (C). In the biological cell, the entry points of the neuron are the nerve endings obtained from the branching of axons that come from other nerve cells, each end being connected by a dendrite. The point of connection is called a synapse. In reality, the neuron endings and dendrites do not touch but are located at a very small distance (50 to 200 angstroms), which is called the synaptic gap. Dendrites come together in bundles, which are their connection to the cell body, called the soma. The output of the soma is a filamentous outgrowth called an axon. It serves to carry electrical energy to the dendrites of another neuron. The condition for the occurrence of an electrical potential depends on the sum of the magnitudes of the stimulating effects that the neuron receives through its inputs. When sufficiently large input stimuli are received, the neuron is activated and generates (fires) an output signal with constant amplitude and variable frequency, i.e. the neuron acts as a threshold function with saturation. This corresponds to a decision making. The high speed of information processing in the biological nervous system is due to the high number of neurons (about 10^{12} in mammals, about 10^{7} in reptiles, and about 10^{5} in flies), and their complex multi-connected topology, which has dynamic behavior.

The mathematical model is much more simple but clearly reflects the main components in the living cell. Each synaptic connection consists of a multiplier that computes the product of a given input with a weight which purpose is to model the strength of

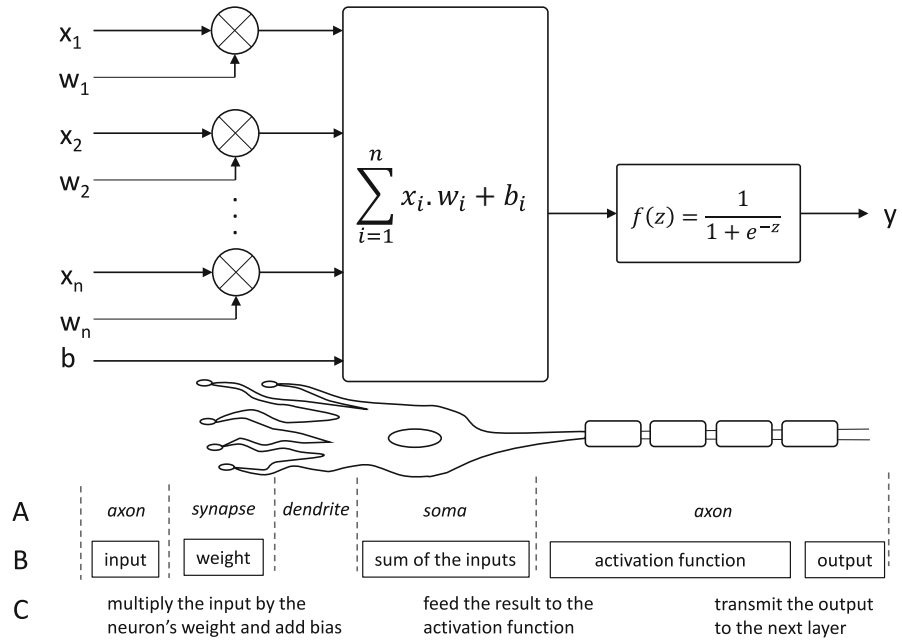


Fig. 1. A model of the artificial neuron along with a functional description: A – biological cell parts description, B – mathematical abstraction equivalent, and C – functions.

the synaptic link. Technically the weights are organized in matrix form and contain the knowledge of the neural network. Their negative or positive values are obtained during the learning process which consists of strengthening or weakening the connections between neurons. The products along with a tuning parameter called the bias, are fed to the cell body which sums them up and feeds the result to the activation function. The latter models the behavior of the axon and may take various predefined implementation forms. It determines the relationship between the neurons of the consecutive layers.

The following conclusions can be drawn from the presented model: (a) the inputs are high-dimensional; (b) the outputs are multidimensional; (c) multiple multipliers are required; (d) one multi-input adder is required; (e) the activation function can be of different types depending on the layer.

In multilayer neural networks, the listed structural elements are multiplied by the number of neurons and can occupy significant hardware resources. Additionally, calculations are performed using signed floating-point arithmetic. When building artificial neural networks using a purely hardware approach, programmable logic circuits of the Field Programmable Gate Array (FPGA) type are most often used. On the one hand, they offer an extremely high level of parallelism and great flexibility, but on the other hand, they do not have built-in floating-point units (FPU) so they do not support the floating-point (FP) standard IEEE-75 [11].

Everything said above leads to the conclusion that instead of having multiple repeating nodes, it is possible to build a single one as shown in Fig. 2. Following the principles

of modularity and regularity [8] instead of generating multiple instances it is possible to use a surrounding (peripheral) infrastructure and organize it in terms of a virtual context.

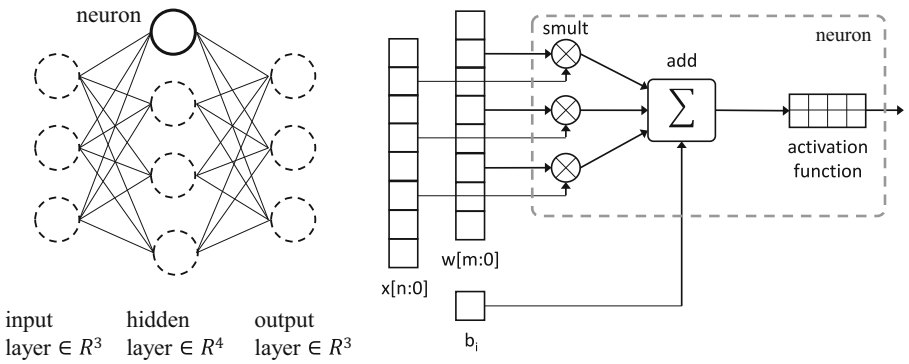


Fig. 2. The artificial neural network consists of multiple repeating nodes (a) that can be substituted with a single neuron (b) which can be utilized multiple times.

Given the three-layer, 7-neuron network from Fig. 2 (not counting the input layer as it just forms the inputs), the processing effort can be computed as 21 multiplications, 28 additions and 7 activations. The numbers for a single neuron unit are seven times less.

The weight matrices (knowledge), the bias (corrective factor that shifts the activation function across the plane), and the activation function (decision making), are structured as arrays. In order to reduce the memory consumption, it is suitable to reduce the precision and process the numbers in signed fixed-point arithmetic [5]. This process in called quantization and can be easily done by performing the shift operation to upscale and downscale the FP number. In [2, 12] it is stated that 8-bit integer multiplies can consume 6X less energy and occupy 6X less area than IEEE 754 16-bit floating-point multiplies, and the advantage for integer addition is improved 13 times in terms of energy and 38 times in terms of area. A transition from 32-bit to 8-bit arithmetic would reduce the model size by a factor of 4, and so there will be a significant reduction in memory. The IEEE 754 standard specifies the 32-bit single precision numbers as a binary number (N_{FP32}) with three fields, namely the sign bit, 8 bits for the exponent (E-127) and 23 bits for the mantissa (M) (1). The exponent is a biased 8-bit unsigned integer, ranging from -126 to $+127$. This is due to the fact that special numbers are represented with -127 (all 0s) and $+128$ (all 1s).

$$N_{FP32} = \pm M \,.b^{E-127} = (-1)^{b_{31}}.2^{(E-127)}.\left(1 + \sum_{i=1}^{23} b_{23-i}.2^{-i}\right) \tag{1}$$

Generally, the quantized 8-bit number N_{q8} can be obtained from the 32-bit IEEE 754 binary number N_{FP32} as shown in Fig. 3 by following (2):

$$N_{q8} = rnd\left(\frac{N_{FP32}}{2.max(abs(N_{FP32}))/256}\right) \tag{2}$$

Fig. 3. IEEE 754 single precision floating point

The denominator is the scaling factor that maps the floating-point dynamic-range to the range [−128, 127], and *rnd* is a rounding function. A possible drawback is that the ANN can lose accuracy because information precision is poorer but, depending on loss factor, the quantized ANN can in fact result in a very minimal loss. This comes at the price of improved latency, memory usage, and power.

3 The Single-Layer Artificial Neural Network

The concept of the single-layer artificial neural network is depicted in Fig. 4.

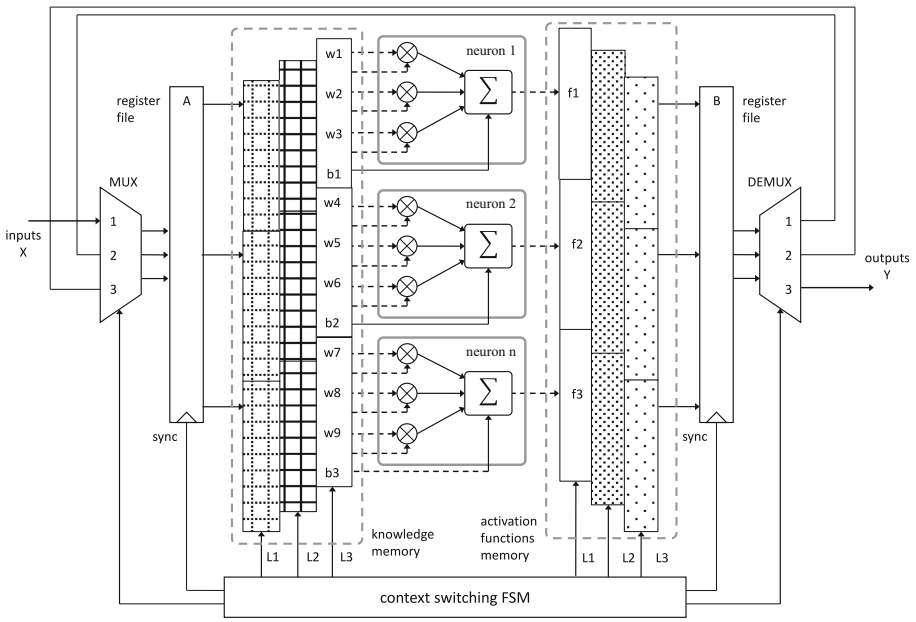
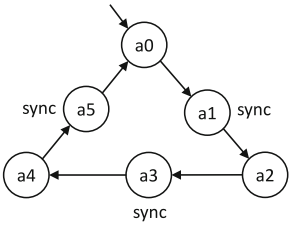


Fig. 4. The single-layer context-switching ANN structure

In a previous research of the authors [7] is has been discussed a model of an artificial neural network with a single hardware layer of processing units. As can be seen from Fig. 2, the ANN consists of a number of repeating structures – neurons, arranged in layers. Each layer is dependent on the data from another layer (in most topologies this is the previous layer). That means instead of multiplying the processing units it is convenient to multiply only the weights, biases and activation functions matrices, and

keep the number of neurons equal to their number in the largest layer since for the unused nodes the matrices will be zeroed. The proposed approach offers a possibility to perform pipelining of the computation chain by switching the parameters set (context) over time. The increase in the processing time between layers is near to zero. The model is simplified and the example has a maximum of three neurons in a layer but it can be easily expanded to more neurons. The vectors of the real inputs and the ones of the subsequent layer outputs are multiplexed and stored one at a time in the input register file (A) which serves the purpose of synchronizing the data transfer. Their values enter the neurons together with the vectors of the knowledge base – the memory arrays that contain weights and biases for the layers L1, L2 and L3. After the data are being processed, the individual neuron firing decision is made based on the associated activation function. Finally, the outputs are stored in another register file (B) and are distributed next with the aid of the output demultiplexer.

The synchronization process is managed by the context-switching finite state machine (FSM) whose directed graph, states encoding, transition, and output tables are given in Fig. 5.



(a)

state	s2	s1	s0
a0	0	0	0
a1	0	0	1
a2	0	1	0
a3	0	1	1
a4	1	0	0
a5	1	0	1

(b)

current state			new state			T2	T1	T0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	0	0	0	1	0	1

(c)

state	A1	A0	B1	B0
a0	0	0	1	0
a1	0	0	0	0
a2	0	1	0	0
a3	0	1	0	1
a4	1	0	0	1
a5	1	0	1	0

(d)

Fig. 5. The context-switching FSM directed graph (a), states encoding (b), state transitions (c), and outputs table (d)

This finite state machine works as a simple counter if taking states a0, a2, and a4 (even states) and as a data transfer synchronizer if taking states a1, a3, and a5 (odd states). The counter functionality takes its outputs directly from the state register and is connected to selector inputs of the input multiplexer and the output demultiplexer. The outputs table (Fig. 5(d) is divided in two halves – the A-side determines the *sync* signal for the input register file and other input-related tasks, while the B-side determines the *sync* signal for the output register file and controls the output data distribution.

The schematic diagram of the synthesized finite state machine device with the aid of T-type flip-flops is shown in Fig. 6. It is in fact a parallel synchronous counter with associated outputs decoding logic. The FSM is of a deterministic synchronous Moore-type. The parallel implementation guarantees that there will be no cumulative delay in the signal propagation chain.

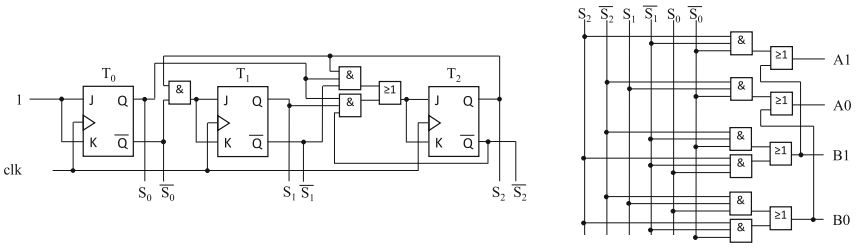


Fig. 6. The synthesized finite state machine (left) and the outputs generation (right)

The general mathematical form of the elaborated single layer neural network is (3).

$$\begin{pmatrix} h_{k1} \\ h_{k2} \\ \vdots \\ h_{kn} \end{pmatrix} = \begin{pmatrix} w_{i1k1} & w_{i2k1} & \cdots & w_{imk1} \\ w_{i1k2} & w_{i2k2} & \cdots & w_{imk2} \\ \vdots & \vdots & \vdots & \vdots \\ w_{i1kn} & w_{i2kn} & \cdots & w_{imkn} \end{pmatrix} \begin{pmatrix} h_{i1} \\ h_{i2} \\ \vdots \\ h_{in} \end{pmatrix} + \begin{pmatrix} b_{k1} \\ b_{k2} \\ \vdots \\ b_{kn} \end{pmatrix}$$
$$= \begin{pmatrix} w_{i1k1}.h_{i1} + w_{i2k1}.h_{i2} + \cdots + w_{imk1}.h_{im} + b_{k1} \\ w_{i1k2}.h_{i1} + w_{i2k2}.h_{i2} + \cdots + w_{imk2}.h_{im} + b_{k2} \\ \vdots \\ w_{i1kn}.h_{i1} + w_{i2kn}.h_{i2} + \cdots + w_{imkn}.h_{im} + b_{km} \end{pmatrix} \quad (3)$$

In this equation h is the single neuron, w is one of the weights, k is the index for next and i is the index for the previous layer of the network. As it can be seen weight and bias matrices can easily fit in two- and single-dimensional arrays respectively. A linear data structure memory array is very convenient for storing the matrices.

4 The Single Hardware-Neuron Network

The artificial neural network discussed above contains a single layer in which computations are performed in parallel. There are no dependencies between individual neurons of this layer. It is therefore possible, at the cost of a small delay, to use a single neuron to perform computations with data sequentially fed to it. For this purpose, it is necessary to add a second state machine to time multiplex the weights coefficients and bias of the neuron's inputs, as well as the activation functions and the final result of its output. The presentation of all the data remains unchanged, in matrix form.

The weights that are associated with each input constitute a multidimensional array. In this array elements in each row correspond to the weights of the respected neuron, the elements in each column are associated with the equally numbered inputs and the depth of the array corresponds to the different layers. The multidimensional array can be transformed into a two-dimensional one as shown in Fig. 7. That means it can fit into a memory and be manipulated using simple addressing arithmetic.

After this step, the array takes the form of the one shown in Fig. 8 which represents an unfolded-in-time structure. This allows the inputs to be multiplexed with the aid of the finite state machine. Each row in this array corresponds to a network layer, and is

controlled by the layer context-switching state machine (Fig. 4). The multiplexing of the weights groups and biases for a neuron is controlled by the time-unfolding state machine.

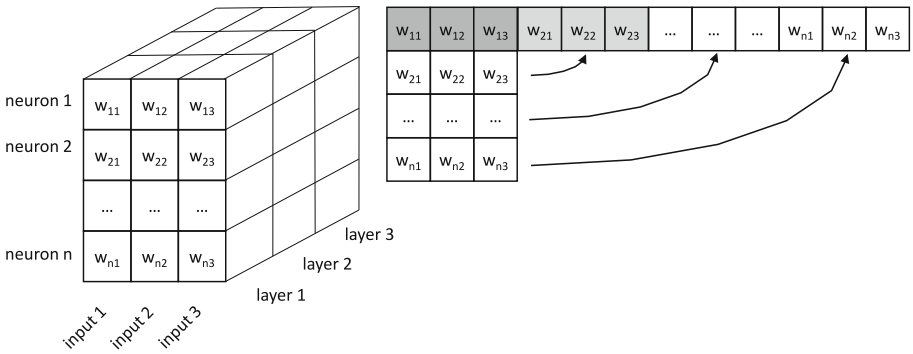


Fig. 7. The three-dimensional array can be converted into a two-dimensional one.

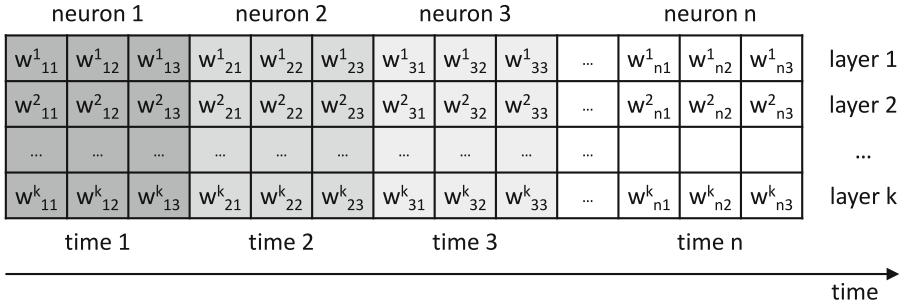


Fig. 8. The two-dimensional representation of the time multiplexed weights matrix. The superscript denotes layer number.

The proposed single hardware neuron architecture is depicted in Fig. 9. There is a single processing unit that involves several multiplications (three in the example) and a single summation at a time. The performance of execution of these operations is dependent on the type of the microarchitecture implementation. The algorithms can be highly optimized for speed, occupied space or power efficiency. The proposed structure also significantly reduces the memory space that is required by the activation function compared with the method proposed in [7]. Moreover, based on the principles of regularity and modularity, the activation function can be implemented in various ways and new functions can be added without needing a redesign of the rest of the circuit. For instance, it is possible to implement the activation function as a look-up table (LUT) which is very fast but not always precise. Or it can be implemented by using some special algorithm such as CORDIC (COordinate Rotation DIgital Computer) or the partial linear approximation of a nonlinear function (PLAN) where the exponent and division are replaced with shift and add operations.

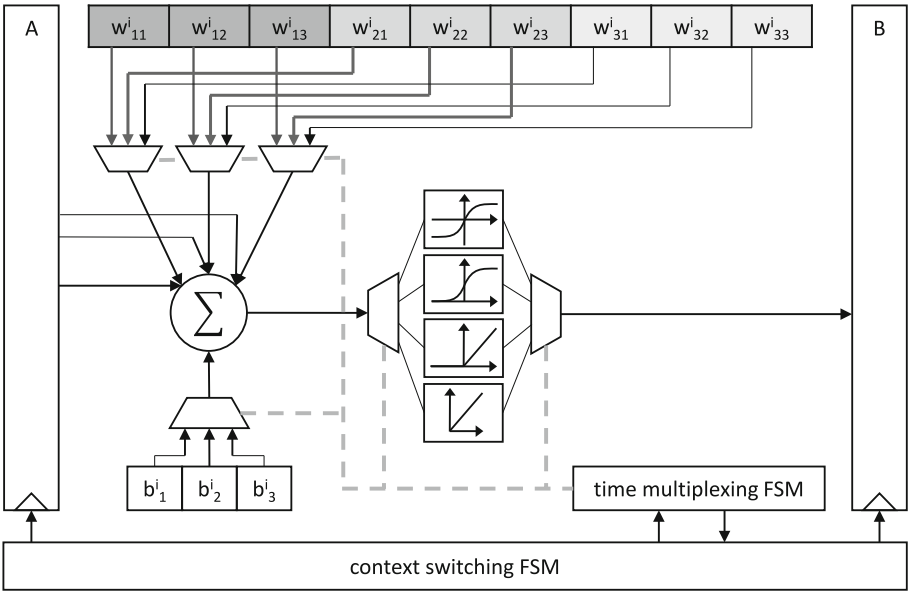


Fig. 9. The single hardware-neuron network

The single hardware-neuron network effectively combines the context switching approach proposed in [7] with the newly proposed time multiplexing mechanism. This suggests that the neural network built using this method will require many times less computing power and energy. A similar structure could be multiplied so that several completely independent artificial neural networks run simultaneously on a single FPGA device which may find myriads of applications in technical fields such as machine learning, robotics and many others.

5 Future Work

The early stage of the present research suggests a huge amount of future work aimed at verifying real networks built in the proposed way, as well as finding new high-performance application areas. It is planned to carry out experiments on the application of the artificial neural network with a single hardware neuron in tasks of functional approximation, simple pattern recognition, and machine learning through the construction of multilayer neural networks. A good candidate where the proposed approach can find a real application, are Graph Neural Networks (GNN), where machine learning algorithms can make useful predictions at the level of nodes, edges, or entire graphs. An irrevocable future task is to conduct a comparative analysis and performance benchmarking with other networks built in a classical way with multiple neurons. The expectations are that the qualities of the proposed network with a single hardware neuron will exceed the indicators of performance, energy efficiency, and area of occupied computing resources, achieved by mass-developed conventional methods.

6 Conclusions

The article presents an innovative author's method for designing artificial neural networks intended for use on devices with limited resources. It combines the switching context methodology for resource sharing, proposed by the authors in previous research, with a novel approach that uses time multiplexing. The latter allows building an ANN using a single hardware neuron which leads to a significant reduction of the processing effort and achieving even better resource utilization. Despite being in an early and immature stage, the suggested design strategy provides a viable mathematical model that deserves to be developed further and opens a quite broad field for scientific research. This method will allow for the design and implementation of highly optimized neural networks that can fit in resource-constrained digital hardware while keeping the performance metrics high. Hopefully, the proposed research will provide research ideas for committed researchers in the field of FPGA-based neural network acceleration.

Acknowledgments. The research paper is written in relation with the agreements between the New Bulgarian University, the China University of Mining and Technology, and the University of Mining and Geology "St. Ivan Rilski" on the subjects "Research and improvement of nodes and elements of the control of mechatronic systems" (MEMF-175/10.05.2023), "Joint Research and Development of key technologies for autonomous control systems", and "Construction of International Joint Laboratory for new energy power generation and electric vehicles".

References

1. Brown, T., Mann, B., Ryder, N., et al.: Language models are few-shot learners. OpenAI (2020). https://doi.org/10.48550/arXiv.2005.14165
2. Dally, W.: High performance hardware for machine learning, cadence ENN summit. NVIDIA Corporation, Stanford University (2016)
3. Eberhart, R., Dobbins, R.: Early neural network development history: the age of Camelot. IEEE Eng. Med. Biol. Mag. **9**(3), 15–18 (1990). https://doi.org/10.1109/51.59207
4. Freeman, W.: Mass Action in the Nervous System. Academic Press (2012). ISBN-13 978-0124120471
5. Gholami, A., Kim, S., Dong, Z.: A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision: Improving the Efficiency of Artificial Intelligence (2012). https://doi.org/10.48550/arXiv.2103.13630
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M. et al.: Generative adversarial nets (2014). arXiv: 1406.2661 [stat.ML], https://doi.org/10.48550/arXiv.1406.2661
7. Gorbounov, Y., Chen, H.: Context-switching neural node for constrained-space hardware. In: Zlateva, T., Goleva, R. (eds.) CSECS 2022. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 450, pp. 45–59. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17292-2_4
8. Harris, D., Harris, S.: Digital Design and Computer Architecture, 2edn. Morgan Kaufmann, Elsevier (2013). ISBN 978-0-12-394424-5
9. Hebb, D.: The Organization of Behavior: A Neuropsychological Theory. Willey, USA (1949)
10. Hopfield, J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. **79**(8), 2554–2558 (1982). https://doi.org/10.1073/pnas.79.8.2554

11. IEEE Std 754-2019, IEEE Computer Society. 2019. IEEE Standard for Floating-Point Arithmetic IEEE STD 754-2019, pp. 1-84, ISBN 978-1-5044-5924-2
12. Jouppi, N., Young, C., Patil, N., et al.: In-datacenter performance analysis of a tensor processing unit. In: 44th International Symposium on Computer Architecture (ISCA) (2017). https://doi.org/10.48550/arXiv.1704.04760
13. Kay, L.: How brains create the world: the dynamical legacy of walter J Freeman in olfactory system physiology. Chaos Complex Lett. **11**(1), 41–47 (2017). PMID: 30686946; PMCID: PMC6344053
14. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25, no. 2 (2012). https://doi.org/10.1145/3065386
15. LeCun, Y., Bottou, L., Bengio, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2323 (1998). https://doi.org/10.1109/5.726791
16. Leijnen, S., Veen, F.: The neural network zoo. In: Conference Theoretical Information Studies, Proceedings, vol. 47, no. 9 (2020). https://doi.org/10.3390/proceedings47010009
17. McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Mathe. Biophys. **5**, 115–133 (1943). https://doi.org/10.1007/BF02478259
18. Medium, Brief History of Neural Networks by Strachnyi, K. https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec. Accessed 21 Mar 2023
19. Minsky, M., Papert, S.: Perceptrons: An Introduction to Computational Geometry. MIT Press (1969). ISBN 0 262 13043 2
20. Murahari, V., Jimenez, C., Yang, R., et al.: DataMUX: data multiplexing for neural networks. In: 36th Conference on Neural Information Processing Systems (2022). https://doi.org/10.48550/arXiv.2202.09318
21. Nurvitadhi, E., Sheffield, D., Sim, J.: Accelerating binarized neural networks: comparison of FPGA, CPU, GPU, and ASIC. In: International Conference on Field-Programmable Technology, Xi'an, China, pp. 77–84 (2016). https://doi.org/10.1109/FPT.2016.7929192
22. Omondi, A.R., Rajapakse, J.C., Bajger, M.: FPGA Neurocomputers. In: Omondi, A.R., Rajapakse, J.C. (eds.) FPGA Implementations of Neural Networks, pp. 1–37. Springer, Boston (2006). https://doi.org/10.1007/0-387-28487-7_1. ISBN-10 0-387-28485-0
23. Poggio, T., Girosi, F.: Networks for approximation and learning. Proc. IEEE **78**(9), 1481–1497 (1990)
24. Pribram, K.: The neurophysiology of remembering. Sci. Am. **220**(1), 73–86 (1969). https://doi.org/10.1038/scientificamerican0169-73
25. Puttagunta, M., Ravi, S.: Medical image analysis based on deep learning approach. Multimed. Tools Appl. (2020). https://doi.org/10.1007/s11042-021-10707-4
26. Ray, P.: A review on TinyML: state-of-the-art and prospects. J. King Saud Univ. – Comput. Inf. Sci. **34**(4), 1595–1623 (2022), https://doi.org/10.1016/j.jksuci.2021.11.019
27. Rosenblatt, F.: The Perceptron - a perceiving and recognizing automaton. Report 85-460-1. Cornell Aeronautical Laboratory (1957)
28. Scarselli, F., Yong, S., Gori, M., et al.: Graph neural networks for ranking web pages. In: IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005) (2005). https://doi.org/10.1109/WI.2005.67
29. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015). https://doi.org/10.1016/j.neunet.2014.09.003
30. Stelzer, F., Röhm, A., Vicente, R., et al.: Deep neural networks using a single neuron: folded-in-time architecture using feedback-modulated delay loops. Nat. Commun. **12**, 5164 (2021). https://doi.org/10.1038/s41467-021-25427-4
31. Wang, C., Luo, Z.: A review of the optimal design of neural keywords: deep learning; deep neural network; FPGA; optimization; hardware acceleration Networks Based on FPGA. Appl. Sci. **12**, 10771 (2022). https://doi.org/10.3390/app122110771

32. Wang, Z., She, Q., Ward, T.: Generative Adversarial networks in computer vision: a survey and taxonomy. ACM Comput. Surv. (2020). ISSN 0360-0300
33. Werbos, P.: Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Committee on Applied Mathematics, Harvard University (1974)
34. Widrow, B., Hoff, M.: Adaptive switching circuits. In: IRE WESCON Convention Record, New York, pp. 96–104 (1960). https://doi.org/10.7551/mitpress/4943.003.0012
35. Wu, L., Cui, P., Pei, J., et al.: Graph Neural Networks: Foundations, Frontiers, and Applications. Springer, Heidelberg (2022). https://doi.org/10.1007/978-981-16-6054-2. ISBN 978-9811660535
36. Yasoubi, A., Hojabr, R., Modarressi, M.: Power-efficient accelerator design for neural networks using computation reuse. IEEE Comput. Archit. Lett. **16**(1), 72–75 (2017). https://doi.org/10.1109/LCA.2016.2521654. ISSN 1556-6056