ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-933

Novel architecture of a digital neuron for FFNN employing special multiplication

Roman Záluský[1], Daniela Ďuračková, Viera Stopjaková, Lukáš Nagy and Vladimír Sedlák

Abstract. This paper presents the design of a new architecture of digital neurons for use in the feed-forward neural networks (FFNN) and their subsequent implementation on a chip. The proposed neuron uses a special type of multiplication realized by AND gate. Comparison of usual ways of implementing digital feed-forward neural networks using fixed/floating point numbers to the novel architecture using the special multiplication was performed. Consequently, the investigated FFNN architectures were implemented into FPGA and ASIC, where the chip area was the main concern. Chip area and other features of both the new neural network architecture and standard NN architectures we compared and evaluated.

1 Introduction

The neural networks represent very important part of the artificial intelligence. Calculation of output activity of huge number of neurons is very demanding on the computer performance. Hardware implementation offers parallel data processing and therefore, the calculation of such a network is very fast. Artificial neuron itself is a complex element, which includes the operations of multiplication, sum and non-linear functions. Using sandard data formats and circuits, only a small number of neurons can be feasibly implemented on the chip to realize the basic operations. Therefore, novel neural network architectures that would use neurons working with operations might significantly help to reduce the network complexity. This would also ensure less chip area overhead and therefore, much larger number of neurons on the chip could be implemented.

In this article, a novel NN architecture that uses a new type of serial multiplication employing a simple AND gate is presented, and the achieved results as well as the main advantages are discussed.

2 Theoretical part

Some systems are very difficult to describe, or are so complex that their description is almost impossible. If we have input data and the required outputs, it is possible to approximate the response of the system. As universal approximators are widely used artificial neural networks that are trained on a given problem. Basic principles of neural networks are referred in [8]- [7].

Operation of a neural network has two modes. The first phase is learning or also called training. In this phase, it is necessary to train the neural network on a given problem. For training, so-called the training data set is used, which is composed of the inputs and the related expected output values. Learned knowledge is stored in the synaptic weight coefficients w and threshold coefficients ϑ. The learning period is followed by the run mode. In this phase, the neural network is able to solve the problem that it was trained for. During the run mode, weights and threshold coefficients are constant. The basic element of the neural network is a neuron. A model of the neuron consists of several inputs x_i, potential of the neuron ξ_i, and the activation function $s(\xi_i)$. The potential is obtained by multiplication of inputs x_i with the corresponding weights w_{ij} and their subsequent summation and addition to the threshold coefficient ϑ_i, as stated in equation 1. The output activity of a neuron is given by processing the neuron potential ξ_i using the activation function $s(\xi_i)$, as stated in equation 2. The activation function has a specific shape and significantly affects the function of the neuron. There are many types of the activation functions refered in [8] [6] [7] [4]. Type of used activation function depends on the network topology and the particular application. For classification purposes and applications, the most suitable type is the sigmoidal function, described by equation 3, where $\mathbf{A}$ and $\mathbf{B}$ are maximum and minimum of the sigmoidal function and α is the slope.

$$\xi_j(x, w, \vartheta) = \sum_{j=1}^{m} w_{ij} * x_j + \vartheta_j \tag{1}$$

$$y_i = s(\xi_i) \tag{2}$$

$$s(\xi) = \frac{B + Ae^{-\alpha\xi}}{1 + e^{-\alpha\xi}} \tag{3}$$

By arrangement of neurons in several layers, one can obtain the multi-layer feed-forward neural network. Layers are divided into the input layer, hidden layers and the output layer. In feed-forward NN, the input data is distributed and processed in only in the direction from input to output. Neurons of one layer are interconnected to the neurons from the next layer. Thus, the outputs activities y_i in the current layer depends only on the output activities of neurons in the previous layer.

3 Proposed architecture

In this section, the design of the new architecture of a digital neuron is described. As already mentioned, the main advantage of neural networks is their parallelism, which can be fully reached through their implementation in hardware. The role of each neuron is to sum weighted input signals and process them using the nonlinear function. The neuron itself performs operations such as multiplication, addition and nonlinear function. Additionally, there are several digital numeric formats for data representation. Data can be represented by fixed-point or floating-point numbers. In the ANN implementation approaches, we focus on the computation speed, the network

[1] Slovak University of Technology, Institute of Electronics and Photonics, Ilkovicova 3, Bratislava, Slovakia, e-mail: roman.zalusky@stuba.sk

complexity and area overhead on the chip. In our research, five different types of neurons that work with fixed point numbers, floating-point numbers and with use of the new special multiplication employing AND gate [1] [3] have been proposed. Mathematical operations can be realized in series or in parallel, where each of these approaches has its advantages and disadvantages. We designed two architectures of a neuron for fixed point numbers data format. The difference between them is in using serial or parallel multipliers. We also designed another two neuron architectures, the first has been optimized and the other using non-optimized neuron. In these experiments, we utilized more possibilities of implementation of the nonlinear activation function.

The advantage of solution with fixed-point number is mainly the simple implementation of mathematical operations. Data was represented by 9-bit direct code, where the most significant bit is a sign and remaining 8 bits represent the numeric value. Such data representation can express numbers from the interval $<-255; 255>$.

Another type of the architecture is based on working with numbers using floating-point (FP) [9] [5]. The format of floating-point numbers is specified in IEEE 754 standard [2]. In addition to the FP format, the standard also specifies the rules for implementing the basic mathematical operations such as addition, subtraction, multiplication, division, remainder after division, square root, comparison ant it defines two basic FP formats. The first format is a single precision FP (32 bits) and a double precision (64 bits). Hardware realization of mathematical operations with numbers in FP formats is rather difficult task, because the format itself is very complex, for example due to large amounts of the number of bits. Complexity of circuits is greatly reduced by using less-bit FP formats. For our purposes, the most preferred format is Microfloat. It uses only 8 bits (1 byte) to store a floating-point number. The range of values in floating point precision in Microfloat format in standardized form is $<-240\ ;\ 240>$.

3.1 Digital neuron using a special multiplication

The main disadvantage of solutions using integers as well as floating-point numbers is the high complexity of units performing mathematical operations. Therefore, we developed a new method for serial multiplication using simple AND gate. In the proposed method of serial multiplication, the potential of the neuron is calculated in one step, because the operations of multiplication and summation are performed simultaneously. This brings a significant simplification of the neuron circuitry resulting in less area overhead.

$$y = a \circledast b \tag{4}$$

$$\xi = \sum_{j=1}^{m} w_i \circledast x_j + \vartheta \tag{5}$$

The method of multiplication using AND gate is based on subsequent multiplication of two numbers bit by bit using a simple 2-input AND gate. Numbers are always in the interval $<0; 1>$, because a given number is part of the specified range. This means that by two 4-bit numbers it is possible to express 16 values. Therefore, a given number will be the n-th of this value. For example, number 5 would be $5/15$. We will refer to the operations of multiplication symbol as $\circledast$ (equation 4). Multiplication operation is performed over a time interval called "time window". Length of the time window is defined as the maximum value, which the number can take. To perform the

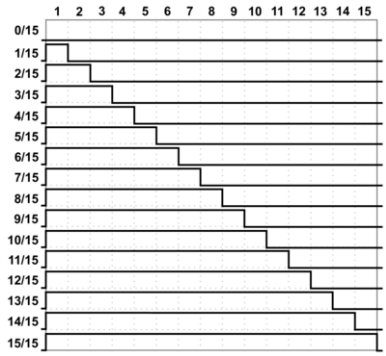

Figure 1. Encoding the number from the beginning of time window

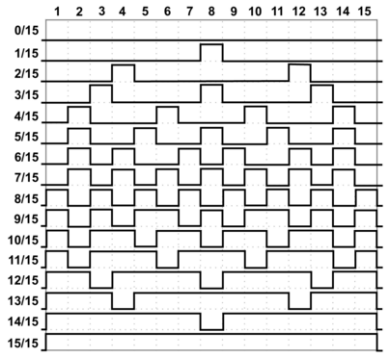

Figure 2. Encoding the number symmetrically around the center of the time window

product of n-bit numbers, a and b are the lengths of the time window $2^n - 1$ time units. For the proper function of multiplication, it is necessary that coefficients are encoded in time. One number has to be encoded in the time from the beginning of the time window (Fig. 1). For example, number 5 would have first five units of time (clock cycles) value of 1 and then value of 0. The other number has to be encoded in the time symmetrically around the center of the time window as shown in Fig. 2. In the multiplication process, it does not matter, which number is encoded one way or another but both numbers must be encoded in different ways. The operation of multiplication is implemented by gating encoded numbers a and b of the individual moments of time window. The product of multiplication is spread over time (Fig. 3). The final result of multiplication is count of the ones in the time window. This multiplication has an effect of natural rounding. In order to be able to work also with negative values, it is necessary to extend the numbers a and b with the signs a_s and b_s. Sign of result y_s is then calculated using the logic function XOR.

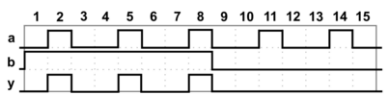

Figure 3. Example of multiplication of numbers a = 5/15 and b = 8/15, the product of multiplication y is 3/15

In conventional architectures, the calculation of the neuron potential is realized in two steps. Firstly, inputs x are multiplied with the

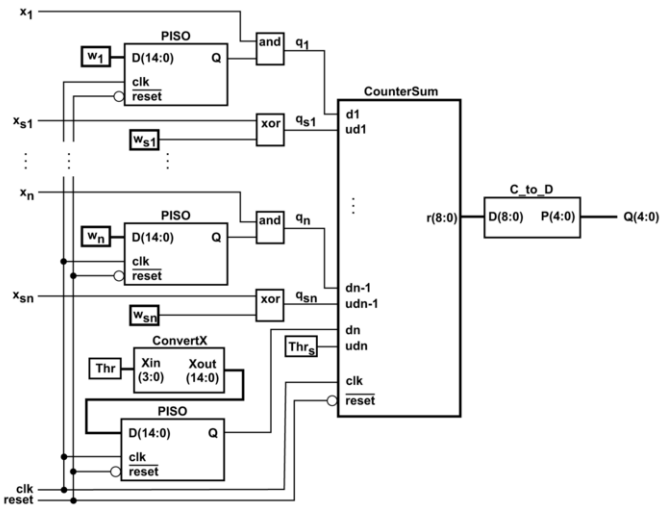

Figure 4. Calculation of the neuron potential

corresponding weight coefficients w. Then, the products of multiplications is subsequently up by use of multi-input adder. In the proposed novel architecture of a digital neuron using multiplication by AND gate, the multiplication of inputs with weights coefficients is performed simultaneously within the duration of one time window. Fig. 4 shows a schematic diagram for the calculation of the potential of a neuron ξ (equation 5). Inputs x_n are encoded from the beginning of time window (Fig. 1). Weight coefficients w_n and the neuron threshold Thr are encrypted symmetrically around the center of the time window, and are permanently stored in the circuit. Inputs x_n are fed with serially encoded data. Weight coefficients w_n and the neuron threshold Thr are converted by $PISO$ (Parallel In Serial Out) register to serial data. To calculate the partial products q AND gates are used. XOR gates are employed to evaluate of signs q_s of the partial products of multiplications. $CounterSum$ summes up the partial products q (take value 0 or 1) within the duration of the time window. With respect to sign of the partial product q_s its value is either added to or subtracted from the state of the counter. Block $CounterSum$ summes together all the partial products simultaneously and therefore, the multiplication of inputs with weights as well as counting of the sum are carried out in one time window. The output of the counter is in the complementary code, which is converted to 5-bit direct code Q by the C_to_D circuit.

Nonlinear activation function is realized by a table. This solution is most suitable for a small number of values because it takes less chip area. The proposed architecture uses 4-bit numbers and one bit to store the sign, so we can express 32 values from the interval $< -15/15\ ;\ 15/15\ >$. In this case, the ROM memory size is 32x5 bits. Function values are calculated for the following parameters of sigmoidal function: $A = -15$, $B = 15$ and $\alpha = 0.3$.

Fig. 5 shows the schematic of a digital neuron using special multiplication by AND gate. Signals x_n are inputs of neuron encoded in time from beginning of time window. Each input x has the corresponding sign x_s. Signal $start$ is used to begin the computation of output activity of the neuron. Circuit sum (Fig. 4) computes the potential of the neuron. The calculation of activation function is realized by circuit $Sigmoid$. Circuit $ConvertX$ encodes the number in direct code from the beginning of time window (Fig. 1), and $PISO$ register converts it to serial data. $TW\_counter$ counts the length of

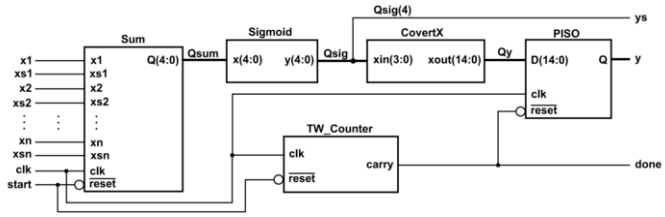

Figure 5. Schematic diagram of a digital neuron employing special multiplication by AND gate with the encoded output

the time window. The neuron output activity is sent serially, and to decode it one time window is needed. Therefore, we optimized the output neurons so that the output activity is in the parallel direct code.

The neuron can be optimized for use in output layer. In the optimized output neurons, circuits for encrypting the output activities have been excluded from the optimized neuron structure. Corresponding signal waveforms during computation of the output activity are shown in Fig. 6. During the first time window, the output activity of the actual neuron is calculated. This output activity, in second time window, feeds inputs of neurons in the next layer. For the optimized neuron, the output activity signal is denoted as Q_{sig}.

It is important to state, the multiplication by the AND gate, which is used in the novel architecture of the digital neuron, has an effect of natural rounding. Nevertheless, the undesired influence of rounding is suppressed during the training process.

3.2 Novel architecture of the feed-forward neural network

The feed-forward neural network consists of neurons organized into multiple layers (Fig. 7). Computation of output activities of the neural network is sequential, layer by layer, while the output activity of neurons in a given layer are calculated in parallel. The output activities of the neuron in previous layer are fed serially to inputs of neurons in next layer. End of calculation in the layer is indicated by a high level of signal $done$. The high level of signal $done$ triggers the computation of the output activities of neurons in next layer. Entire calculation of the output activities of the whole neural network takes as many time windows as many layers the neural network has. In the optimized neural network, the output layer is composed of the optimized neurons. Neural network of the proposed architecture using a special multiplication by AND gate needs dedicated circuits for en-

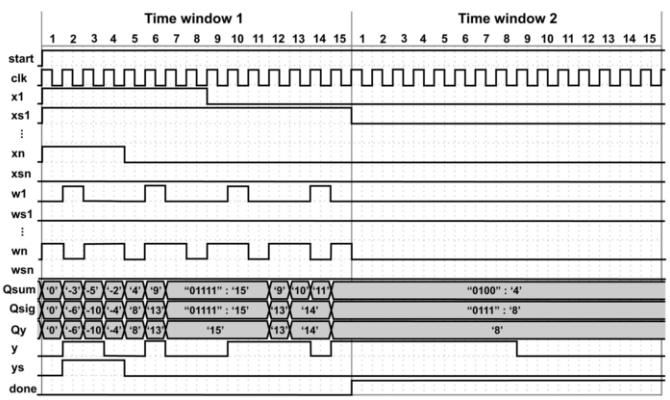

Figure 6. Signals of the digital neuron with the special multiplication during computation of the output activity in time

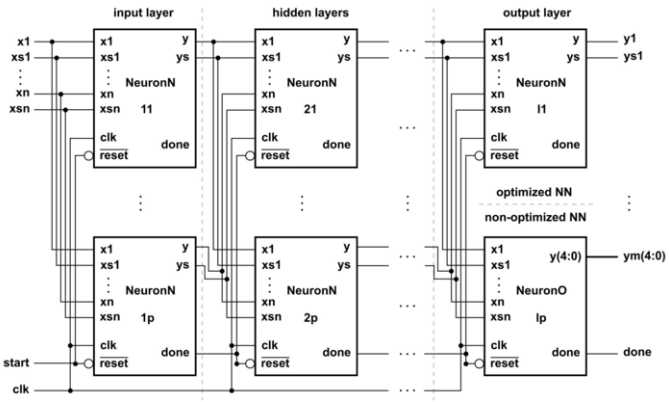

Figure 7. Schematic of novel architecture of feedforward neural network

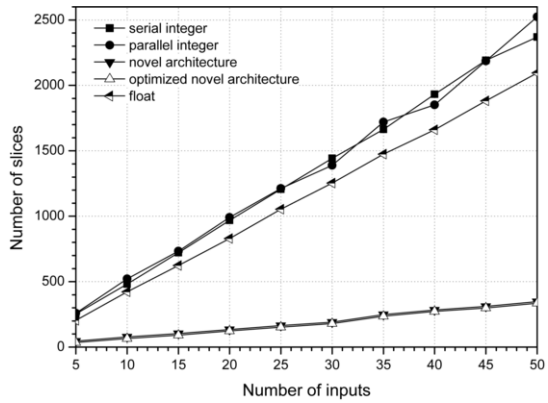

Figure 8. Comparision of chip area for neuron versus the number of inputs

coding the network inputs and decoding the output activities. Decoding the network output activities takes one time window, and thus, the total computation of the output activities takes $l + 1$ time windows, where l is the number of layers forming the neural network. Output activities in the optimized neural network are represented in the direct code and therefore, no additional circuits for decoding them are needed. This will make the computation of output activities in one time window shorter.

4 Achieved results

All the architectures were described by VHDL language and synthesized for three FPGA Xilinx Spartan 3 chips of different sizes, as listed in Table 1. FPGA chip consists of configurable logic blocks (CLB). Each CLB contains four slices, and each slice contains two look-up tables and two flip-flops, multiplexers and logic gates. The chip area is determined mainly by the number of slices needed to implement the circuit. The number of bits for each input of a neuron varies with the respective NN architecture. The proposed NN architecture employing the special multiplication uses 5-bit inputs, while the architecture working with floating-point numbers uses 8-bit inputs, and the integer-based architecture contains neurons with 9-bit inputs.

Table 1. List of FPGA chips

Type	CLBs	Slices
xc3s1000	1920	7680
xc3s1500	3328	13312
xc3s2000	5120	20480

We compared the chip area of the neuron circuits for increasing number of inputs in the range from 5 to 50 with step of 5. The obtained results for the proposed neural network architectures are shown in Fig. 8. For the implementation to FPGA chip, both neuron types of the novel architecture need approximately 7 times less number of slices. Moreover, the digital neuron, which uses the optimized novel architecture consumes 10 slice less than that of the non-optimized architecture. This is because the neuron optimization is performed only on its output and does not affect the calculation of the potential of a neuron.

Next, the comparison of chip area for feed-forward neural networks versus the number of hidden layers (with constant number of

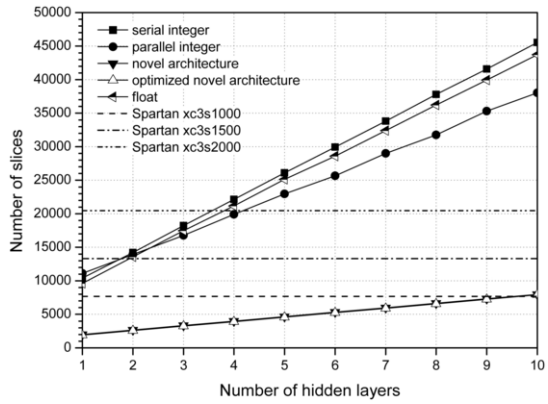

Figure 9. Dependance of FFNN chip area on the number of hidden layers

hidden neurons) has been carried out. The input layer consisted of 20 input neurons, each hidden layer was formed by 10 hidden neurons, and output layer contained 6 output neurons. Number of hidden layers was varied in the range from 1 to 10 with step of 1. Obtained results are shown in Fig. 9. Dependency of chip area consumption on the number of hidden layers of neural network is linear. This is due to only the increasing number of neurons and connections between neurons, without changing complexity of the neurons. The smallest number of slice blocks needed to implement the neural network in FPGA chip was achieved for both types of the novel architecture, which needed on an average 5 times smaller slice number of blocks than the other three architectures. In the smallest type of FPGA chip Spartan xc3s1000 (dashed line), it can be implemented only a neural network with novel architecture (non-optimized and optimized) of complexity up to 9 hidden layers. To implement the novel architecture of the neural network containing 10 hidden layers it is necessary to choose a larger type of FPGA chip, e.g. Spartan xc3s1500 (dash dot line). On the other hand, none of the other three regular NN architectures can be implemented in the smallest FPGA chip Spartan xc3s1000. Medium size FPGA chip Spartan xc3s1500 would be able to implement a neural network, which contains only one hidden layer. In the largest type of FPGA chip Spartan xc3s2000 (dash dot dot line) it is possible to implement classical neural networks with up to three hidden layers.

For further comparison of the chip area depending on the neural network topology, we varied the number of neurons in hidden lay-

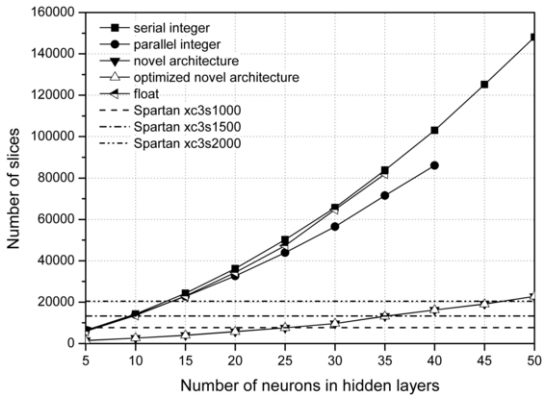

Figure 10. Chip area of FFNNs versus the number of neurons in hidden layers

ers, while keeping the number of hidden layers constant. Each neural network consisted of two hidden layers. The input layer consisted of 20 input neurons, both hidden layers contained hidden neurons in the range from 5 to 50 with step 5, and the output layer was composed of 6 output neurons. The chip area as a function of the number of neurons in the hidden layers is shown in Fig. 10. The obtained dependences are exponential. This is due to the increasing number of neurons, connections between neurons, and also the enhanced complexity of neurons for the increasing number of inputs (Fig. 8). The proposed novel neural network architecture requires on an average of 5 to 6 times less slices to be implemented in an FPGA chip in comparison to the other three architectures. To the smallest FPGA chip Spartan xc3s1000 (dashed line) can be implemented a neural network of the novel architecture with two hidden layers and 25 hidden neurons in each hidden layer. Medium size FPGA chip Spartan xc3s1500 (dash dot line) can implement a neural network of the novel architecture containing two hidden layers with 35 hidden neurons. Finally, into the FPGA type xc3s2000 (dash dot dot line), a neural network of the architecture with two hidden layers and 45 hidden neurons in each hidden layer can be implemented. The other three architectures of neural networks can be implemented into the smallest FPGA chip xc3s1000 only in the complexity containing two hidden layers with 5 hidden neurons in each hidden layer. To the largest type of FPGA chip xc3s2000 can be implemented a neural network with two hidden layers of 10 hidden neurons in each hidden layer.

Neural networks working with integers and using serial multipliers take a larger number of slices because they use both combinational and sequential circuits. Neurons working with floating-point numbers use Microfloat format, since it is the simplest FP format. Taking this fact into account, the novel neural networks require less chip area compared to architectures using integer or float numbers. This is due to simpler neurons needed for the novel architecture. Other reason is that the new neural network architecture includes 1-bit nets between neurons, through which the signals are distributed serially, while the other NN architectures using multi-bit signals (8 and 9-bits) work with parallel data transfer.

In order to evaluate the proposed network architecture also in alternative hardware implementation and be able to compare the chip area to other works, the synthesis to a standard cell based ASIC has been performed as well. Table 2 show the synthesis results of previously described network architectures. Synthesis has been carried out for standard CMOS general purpose technology TSMC 90 nm. Table contain information about the total number of used cells, the overall area consumption and the time required for successful synthesis. The obtained results also prove that the proposed NN architecture exhibits significant improvement in silicon area consumption as well as the synthesis time. The new architecture of neural network, which consists of two hidden layers with 10 hidden neurons in both hidden layers and 6 output neurons occupies approximately 0.15 mm^2.

Table 2. Synthesis results for 90 nm CMOS technology

2x10	90 nm CMOS process		
	Cell count	Area [mm^2]	Synth. time [min]
Serial integer	62 900	0.6476	170
Parallel integer	94 784	0.5861	170
Float	29 948	0.1804	60
new non-optimized	16 063	0.1508	15
new optimized	15 604	0.1469	15

Table 3. Comparison of computation time of neural networks with two hidden layers

Architecture	f_{max} [MHz]	t_{comb} [ns]	t_{calc} [ns]
Serial integer	79.051	101.74	405.34
Parallel integer	-	124.9	124.9
Float	-	821.54	821.54
new non-optimized	37.477	-	1609.8
new optimized	37.477	-	1200.7

Finally, the calculation time t_{calc} of output activities of the neural networks was investigated. In this experiment, all neural networks contain two hidden layers. The values of the maximum frequency f_{max} of clock signal were obtained through synthesis of VHDL description of each architecture. For architectures working with combinational circuits, the delay of the combinational logic t_{comb} was measured and evaluated from the synthesis. In Table 3, results for the maximum frequency, the logic delay, and the calculation time of output activities for each neural network are presented. The fastest neural network is a network operating with integers using parallel multiplier, which uses only combinational circuits. The neural network operating with floating-point numbers exhibits about 6 times longer calculation time. This is mainly due to the feedback circuitry necessary for adjusting mantis and exponents. Finally, the novel NN architecture proposed in this work can operate at the maximum frequency of 37,477 MHz, which is about half the frequency achieved for the neural network operating with integers using serial multipliers. At this frequency, the length of time window is about 400 ns. Thus the computation time of output activities for the non-optimized and the optimized neural network takes 1610 ns and 1200 ns, respectively.

The learning efficiency as well as the generalization process were evaluated on solving the XOR problem. Firstly, the dependence of the number of iterations in the training process, needed for various NN topologies, was examined. Here, the number of hidden layers was increased in the range from 1 to 3, while two and three hidden neurons were used in each hidden layer. The learning coefficient was $\lambda = 0.01$ for all networks. The maximum error varies for different neural networks since they work with different range of values. For the neural network working with floating-point numbers, the maximum error $E_{max} = 0.1$. For the novel NN architecture as well as the neural network using integers, the maximum error $E_{max} = 1$. Ten runs of the learning process were carried out and the average value

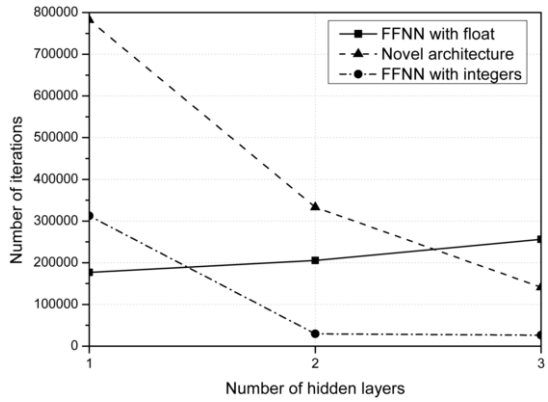

Figure 11. 2 hidden neurons in each hidden layer

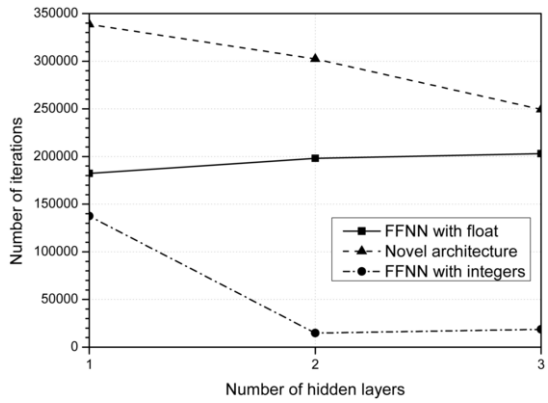

Figure 12. 3 hidden neurons in each hidden layer

was evaluated. Achieved results are shown in Fig. 11 and Fig. 12. One can observe that the network topology (complexity) has no significant effect on the learning speed for the neural network working with floating-point numbers. The novel NN architecture and the architecture using integers achieve higher learning speed for an increased number of hidden layers and hidden neurons. The novel architecture exhibits the slowest learning in comparison to other architectures. Generalization ability versus the network topology is presented and discussed in the next section on the recognition of characters of Dactyl alphabet.

5 CONCLUSION

A novel architecture of a digital neuron, which uses the special multiplication by basic AND gate is proposed, designed and implemented. The aim was to achieve the smallest chip area in hardware implementation. Consequently, the new architecture of a digital neuron was compared to neurons working with integers and floating-point numbers. The novel architecture takes approximately about 2/3 less chip area than the second simplest conventional network architecture. In comparison to the most complex architecture, the proposed novel neuron hardware is roughly 6 times smaller.

The achieved results prove that the proposed digital neuron and also the new architecture of feed-forward NN offer significant reduction of chip area, if implemented in hardware. This is rather substantial advantage from the application point of view, since many diverse applications might profit from on-chip systems using artificial neural networks.

ACKNOWLEDGEMENTS

This work was supported in part by Competence Center for SMART Technologies for Electronics and Informatics Systems and Services (ITMS 26240220072), funded by the Research&Development Operational Programme from the ERDF, by the Ministry of Education, Science, Research and Sport of the Slovak Republic under grants VEGA 1/0987/12 and VEGA 1/0823/13.

REFERENCES

[1] R. Zalusky E. Raschman and D. Durackova, 'New digital architecture of cnn for pattern recognition', *Journal of ELECTRICAL ENGINEERING*, **61, NO 4**, 222–228, (2010).
[2] W. Kahan, 'Ieee standart 754 for binary floating-point arithmetic', *Elect. Eng. and Computer Science*, (1997).
[3] E. Raschman and D. Durackova, 'Area chip consumption by a novel digital cnn architecture for pattern recognition', *Artificial Neural Networks - ICANN 2009, 19th International Conference*, 363–372, (2009).
[4] T. N. Prabakar S. Hariprasath, 'Fpga implementation of multilayer feed forward neural network architecture using vhdl', *Computing, Communication and Applications (ICCCA), 2012 International Conference*, 1–6, (2012).
[5] D. Mic A. Buchman S. Oniga, A. Tisan, 'Optimizing fpga implementation of feed-forward neural networks', *Optimization of Electrical and Electronic Equipment, 2008. OPTIM 2008. 11th International Conference*, 31–36, (2008).
[6] P. Sincak and G. Andrejkova, *Neural networks (The engineering metodology) part 1*, Elfa s.r.o., Kosice, 1996.
[7] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*, Academic Press, Boston, 2009.
[8] I. Farkas A. Kral J. Pospichal V. Kvasnicka, L. Benuskova and P. Tino, *Introduction into the neural networks (In Slovak)*, IRIS, Bratislava, 1997.
[9] R. Yates, 'Fixed-point arithmetic: An introduction', *Digital Sound Labs*, (2001).