An Analysis of FPGA Hardware Platform Based Artificial Neural Network

IOP Conf. Series: Journal of Physics: Conf. Series **1201** (2019) 012009 doi:10.1088/1742-6596/1201/1/012009

An Analysis of FPGA Hardware Platform Based Artificial Neural Network

F W Wibowo[1]

[1] Informatics Department, Universitas Amikom Yogyakarta, Jalan Ring Road Utara, Condong Catur, Depok, Sleman, Yogyakarta 55283, Indonesia
Email: ferry.w@amikom.ac.id[1]

Abstract. The artificial neural networks (ANN) algorithm is a mathematical model of a network by applying neurons and usually, it is represented as a directed graph with vertexes and edges. This algorithm is a paradigm of information processing to describe arbitrary functions and inspired by the biological nervous systems. This algorithm can learn a non-linearly separated set of outputs. A much software-based application has implemented this algorithm but the challenge in the hardware-based is still possible to do in mapping the output into binary values. This paper addressed on the platform of the hardware-based the ANN algorithm. The field programmable gate arrays (FPGAs) is closely related to this platform. The FPGAs have achieved a successful penetration in different multi-disciplinary domains.

1. Introduction

The artificial neural networks (ANN) algorithm is a paradigm of information processing to describe arbitrary functions and which is inspired by the biological nervous systems. This ANN algorithm consists of neurons system that is connected with the synapses. The neuron collects signals from the dendrites and sends the spikes of the electrical activities out via an axon with splitting into thousands of branches. The neuron will act when the exciting activities surpass the inhibitory activities. Learning changes the synapses effectiveness and at the end, the synapses convert the activities into exciting or inhibiting activities. The illustration of the nervous systems then adopted into a computational model. In the year 1890, William James had defined a neuronal process of learning. In the year 1943, this ANN model had been proposed by McCulloch and Pitts by using their knowledge about the neurology. There were known as the simple logic functions of α AND β and α OR β. In the year 1950s, there were several researchers who had tried to model the biological behavior, they are Farley and Clark; Rochester, Holland, Haibit, and Duda. Donald Hebb and the International Business Machines (IBM) research group had done the earliest simulations in the year 1954. In the year 1958, Frank Rosenblatt had discussed the perceptron with the three-layer systems which are input nodes, association layers, and output nodes. This discussion provided an ability in learning of connection or association a given input to a random output and it introduced the first concrete neural model. The learning process is meant to reduce the error value that can be understood as the difference between the target and output values from the structure of learning. The ANN learning methods are quite robust to noise in the training data so that the long training times are acceptable in this case. It depends on the factors of the weight numbers in the network. The association unit extracts the features from the inputs, this function has a role to select the process result using such a threshold with the weighted sum of input. While the output is weighted and associated. In the year 1960 until 1962, Bernard Widrow and Marcian Hoff had proposed an Adaline and least mean square (LMS) learning rule to minimize the error over all training patterns. Thus in the year 1969, Minsky and Papert had shown that a single layer perceptron could not learn the exclusive

OR (XOR) of two binary inputs in the other word could be said that the perceptron has severe limitations e.g. able to form linear discriminate functions only and the most functions were more complex. In the year 1974, Werbos had proposed the method of backpropagation learning using the three layers of neurons i.e. input, hidden layer, and output. This approaching has better learning rules for generic three-layer networks and this work had been continued until the 1980s. The nodes hidden layer allowed combinations of linear functions. The non-linear activation functions showed properties that approach to the real neurons which the output varies continuously but not linearly and differentiable using sigmoid. So that the non-linear of the ANN classifier was possible to do. However, in that time, there was no learning algorithm to set the weights of multi-layer networks because the weights shall be set manually. In the year 1982, Hopfield had proposed his thought about a neural network that it did not same with the neurons in the multi-layer perceptron (MLP). His proposal contained only one layer whose neurons are fully connected with each other. The Boltzmann machine had been influenced by both the Hopfield network and the MLP. In that year also another network model of the self-organizing map (SOM) had introduced by Kohonen. The SOM originated from the learning vector quantization (LVQ) network that was highlighted his thought in the year 1972. In the year 1985, the multi-layer nets had used a back-propagation and in the year 1986, the PDP research group has used as a multi-disciplined approach. In this year, Rummelhart and Mclelland introduced a general back-propagation for the MLP. During that time was found the solution of the multi-layer ANN's weight that was set update e. The concept was simply using the global error to be backward propagated to the network nodes and the weights were modified proportional and the most significant has broken problem by using the concept of the ANN learning algorithm. This concept then was known as the back-propagation. In the year 1988, Broomhead and Lowe introduced the radial basis function (RBF) networks although its basic had been developed 30 years ago with the name of the potential function method. Later in the year 1990s, this ANN method has been widely implemented in many fields. Since then, this model has remained active and many hybrid algorithms have been proposed for the neural information processing whether software-based and hardware-based.

This paper focusses on the platform of the hardware-based the algorithm of the artificial neural network. One of the platforms of the hardware-based is a field programmable gate arrays (FPGAs) which is closely related to the reconfigurable computing (RC). In embedding algorithms to the FPGAs has many factors to be considered such as the design that affects speed and area. The speed aspect depends on the time used provided by the paths of the route and logic while the area depends on the consumption of the components of the FPGAs utilized. It is very possible to embed the algorithm into the FPGAs as one of the hardware platforms but the requirements of the hardware design must be familiar and known by the hardware programmer/designer [1]. So that, the FPGAs have achieved successful penetration in different fields. The ANN can be depicted as a directed graph which consists of vertexes and edges. It represents simple processing units those communicate by sending signals to each other through a number of the weighted connections.

2. Related Works

The field programmable gate arrays (FPGAs) are very high logic capacity that allows to implement digital hardware. The ability of the FPGAs can be modified its operation during runtime so it allows being customized as per needs of the user by reconfiguration or programming. The advantages of the runtime reconfiguration in the implementation of the FPGAs are reducing the power consumption, reducing time to reconfigure, flexibility, obsolescence avoidance, application portability, and the hardware reuse [2]. The hardware designer can use the same FPGAs more than design because it has a fast digital design and good prototypes. Its features allow being suitable for applying parallel architectures. Two main architectures those are offered by the FPGAs are fine-grained and coarse-grained. The fine-grained consists of a large amount of the small logic blocks so it has good utilization while the course grained consists of the smaller number of larger and more powerful logic blocks so it is faster because of easy routing. So that the selection of the FPGAs in the implementation has important position besides the design entry of the hardware description language (HDL) [3]. Partially configurable circuits can be implemented in the FPGAs using modular designs with simple circuits only. It will allow

testing different possible combination when it uses a partial reconfiguration so this makes easy to track errors or warnings.

The FPGAs can be reconfigured using the code of the Very High-Speed Integrated Circuit Hardware Description Language (VHDL). It has been used to reconfigure the FPGAs to implement a feed-forward ANN [4]. The implementation of the ANN design using the FPGAs has employed the read-only memory (ROM) and random access memory (RAM) for efficiency and flexibility. The weights have been set in the ROM, whereas the input and the layer output has been set applying RAM. So it can reduce the complexity of the design of the activation function. If the function used for designing is the sigmoid, a method in implementing this design can utilize a linear log sigmoid function [5]. The design of the activation function could implement by using a multiplexer (MUX) or ROM/RAM. The selected value for making a function it is significant to do.

3. Methodology

The basic function of a neuron is to sum inputs and provide an output that is given by sum which is greater than the threshold. The ANN node produces an output by multiplying each component of the inputs by the weight of its connection. Then, sum all weighted inputs and subtracts the threshold value to obtain the total weighted input. After that, transform the total weighted input into the output using a function of the activation. This diagram structure can be depicted in Figure 1.

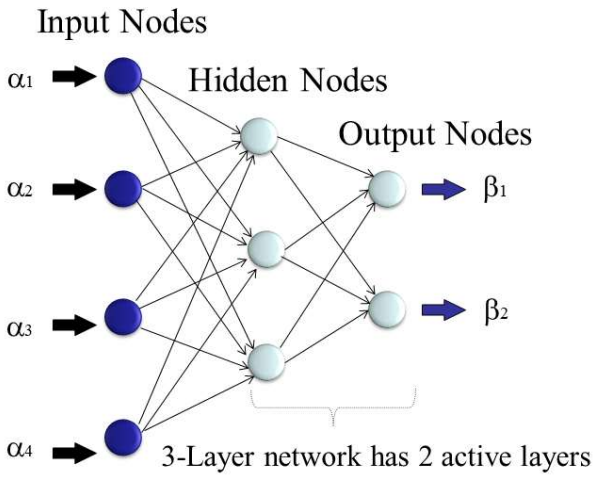

Figure 1. Diagram structure of the ANN algorithm.

There are 3 things that should be noted i.e. structure of the network, the structure of each node, and weights on each of the connections. These things determine the flow of the ANN results. The output β of the ANN is a function of the input α which is affected by the weights ϖ and the activation functions $\xi(\chi)$ as shown in Figure 2.

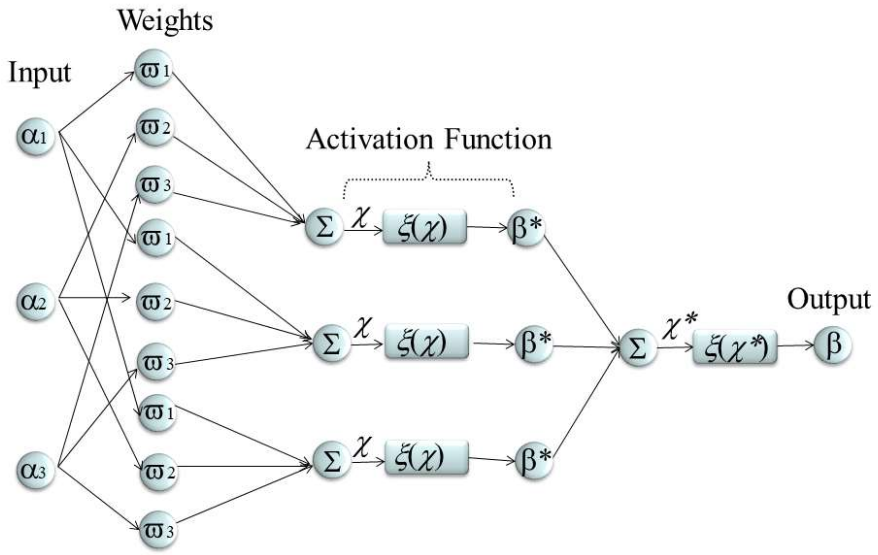

Figure 2. Illustration of the ANN algorithm.

The transfer functions can be categorized into three types i.e. linear, threshold, and non-linear. The linear function has an output that is proportional to the total weighted input, the threshold function has an output that is set at such the total weighted input as greater than or less than some threshold values, whereas the non-linear function has an output that varies continuously but not linear as the input changes. Basically, there are two methods of the ANN i.e. the ANN forward propagation and the ANN backward propagation. In the ANN forward propagation, calculating is done from the input layer to the output layer and for each neuron, there are two calculations which are calculating the weighted average of input and activation function. The perceptron learning algorithm steps can be written as below,

1. Initialize the weights,
2. Determine the network architecture and the target output,
3. Compute the output as equation 1 with ξ is a function,

$$\beta = \xi\left[\sum_{i=0}^{n} \varpi_i \alpha_i\right] \tag{1}$$

4. Update the weights ϖ as equation 2 with ι is the present state,

$$\varpi_i(\iota+1) = \varpi_i(\iota) + \Delta\varpi_i \tag{2}$$

5. Repeat the step from point 2 until it has obtained the acceptable level of error.

In the year 1960s, Widrow and Hoff had proposed the delta rule, in this case, it is implemented for making a weight modification. It is necessary adjusted to minimize the error after each iteration. This equation can be shown in equation 3.

$$\Delta\varpi_i = \eta\varepsilon\alpha(\iota) \tag{3}$$

Where the η is the learning rate which has range values of $0 < \eta <= 1$ whereas the ε is the error signal which is a difference between the desired output and the network output. If the learning rate is varied, it will cause vary training time. The example of the output activation level ϑ is shown in equation 4.

$$\vartheta(\alpha)=\begin{cases} 1; & \alpha \geq \frac{1}{2} \\ \alpha; & 0 \leq \alpha \leq \frac{1}{2} \\ 0; & \alpha \leq 0 \end{cases} \tag{4}$$

While the concept of the back-propagation is the perceptron-like but the error calculation ε is related to the difference between the target output τ and the actual output β as shown in equation 5.

$$\varepsilon = \frac{1}{2}\sum_{i=0}(\tau_i - \beta_i)^2 \tag{5}$$

However, in the back-propagation, the change value of the error is the important thing to feedback via the network. This algorithm performs gradient descent in weights to obtain the minimum level of the error. The gradient is given by equation 6.

$$\frac{\partial \varepsilon}{\partial \varpi_{ij}} \tag{6}$$

This shows that rate will change along with the error changes as the weight change. So this method can be approached using a generalized delta rule with considering the sigmoid activation function as shown in equation 7.

$$\Delta \varpi_{ij} = -\eta \frac{\partial \varepsilon}{\partial \varpi_{ij}} \tag{7}$$

Where the η is the learning rate and the $\partial\varpi_{ij}$ is the weight from the node *i*-th to node *j*-th while for the sigmoid activation function considered as the output of the node is shown in equation 8.

$$\xi(\chi_j) = \frac{1}{1+e^{-\chi_j}} \tag{8}$$

Where the ξ is the function and the χ_j is the totaled weighted input of node as shown in equation 9.

$$\chi_j = \sum_{i=0}^{j} \varpi_{ij}\beta_i \tag{9}$$

In calculating of the rate of error and weight must be considered the fast error changes as weight coming into node *j-th*, output of node *i-th* in the previous layer, the output of the node *j-th*, and total input to node *j-th* are changed. It can be summarized the back-propagation algorithm as written below,

1. Initialize the weights,
2. Determine the network architecture and the target output,
3. Compute the output as equation 10,

$$\beta_j = \xi\left[\sum_{i=0}^{n} \varpi_{ij}\beta_i\right] \tag{10}$$

4. Update the weights as equation 11,

$$\varpi_{ij}(\iota+1)=\varpi_{ij}(\iota)+\Delta\varpi_{ij} \tag{11}$$

Where ι is the present state. For $\Delta\varpi_{ij}$, it can be known from equation 12,

$$\Delta\varpi_{ij}=\eta\beta_j\alpha_i=-\eta\frac{\partial\varepsilon}{\partial\varpi_{ij}}=-\eta\varepsilon\omega_{ij} \tag{12}$$

For output nodes can be shown in equation 13 with the κ is a linear discriminate function,

$$\beta_j=\varepsilon\alpha_j^*=\varepsilon\kappa_j\alpha_j(1-\alpha_j)=\alpha_j(1-\alpha_j)(\iota_j-\alpha_j) \tag{13}$$

For hidden nodes can be shown in equation 14,

$$\beta_i=\varepsilon\alpha_i^*=\varepsilon\kappa_i\alpha_i(1-\alpha_i)=\alpha_i(1-\alpha_i)\sum_j\varepsilon\alpha_j^*\varpi_{ij} \tag{14}$$

5. Repeat the step from point 2 until it has obtained the acceptable level of error.

4. Result and Discussion

The ANN implementation in the hardware design of FPGAs can be classified into three approaches which are a small scale approach (SSA), large scale approach (LSA), and layer structure approach (schematic). The small scale approach is simple and easy to be understood. The large scale approach is used to more generic approach that is suitable with the large scale networks. The design of the SSA and LSA is reconfigured using FPGA while the schematic is reconfigured using the schematic design [6]. Another way to reconfigure the FPGAs implementing ANNs could use components in defining the main components in the VHDL code. The main components are a multiplication component, addition component, and the activation function component. The activation function component can vary in designing using the VHDL code [7]. The synthesis results of both SSA and LSA have provided same numbers of the area and speed that is using Xilinx FPGA from the family of Artix7 XC7A100T as shown in Table 1.

Table 1. Comparison of the area and speed of the FPGA implementation of the ANNs first layer with three neurons, three inputs, and weights per neuron.

Bit	Area					Speed	
	Number of slice registers	Number of slice LUTs	Number of fully used LUT-FF pairs	Number of Bonded IOB	Number of DSP48E1s	Logic (ns)	Route (ns)
4	36	270	9	45	0	2.964	4.683
8	64	158	8	89	9	5.169	2.498
16	128	310	16	177	9	5.905	2.527
32	256	608	32	353	36	10.240	2.459

The different design of the SSA and LSA lies on the usage of the package design in the VHDL code. For the ANN design of 4-bit, number of DSP48E1s is not consumed. It has been utilized when the design of the ANN using above 8-bit. So if it is assumed that the parameter of the area and the speed as shown in Table 1, the comparison of area and speed can be shown in figure 3.

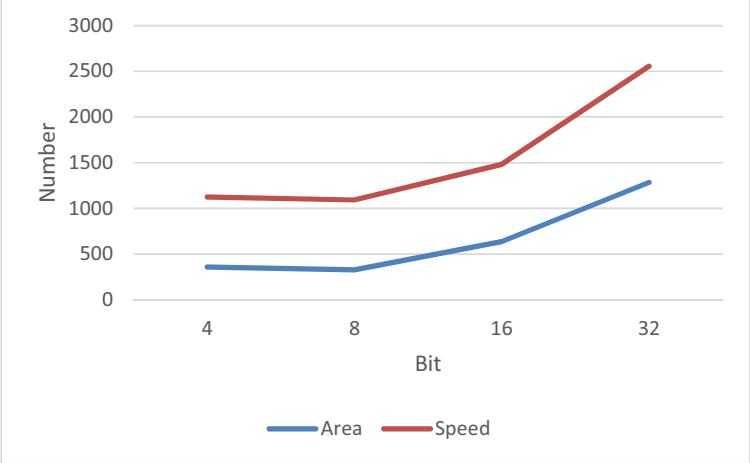

Figure 3. Diagram of the hardware-based the ANN.

From Figure 3, it can be seen that the area and the speed of the design are almost linear increasing as long as an increasing bit. The unit of the area is a unit number of FPGA's component consumption meanwhile the unit of the speed is 10^{-11} second. The speed unit is larger in order to be seen the difference with the value of the area. The design diagram of the ANNs using the FPGAs can be shown in Figure 4.

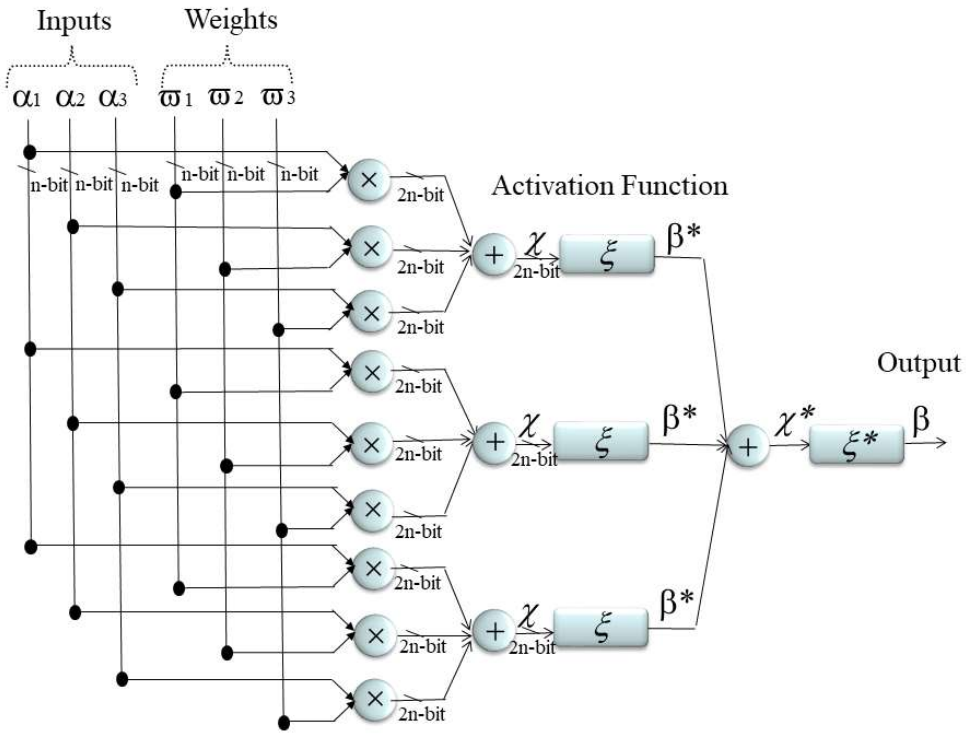

Figure 4. Diagram of the hardware-based the ANN.

Each input is associated with the weights in multiplication. For the implementation of multiplication can be applied an appropriate multiplication method because for each increasing bit input has a different efficiency [8]. The design of the ANN using ROM and RAM has a cost consumption but it is efficient

and the otherwise, the design using figure 5 is a low-cost consumption but it needs a complexity especially designing the activation function. For designing the activation function could be implemented of the sigmoid or others with implementing the look-up table (LUT) that in this case is designed using ROM or it can also be designed using RAM for flexibility [9]. The complexity of the design can be very tricky. Meanwhile in designing the backward ANNs could be designed as shown in Figure 5.

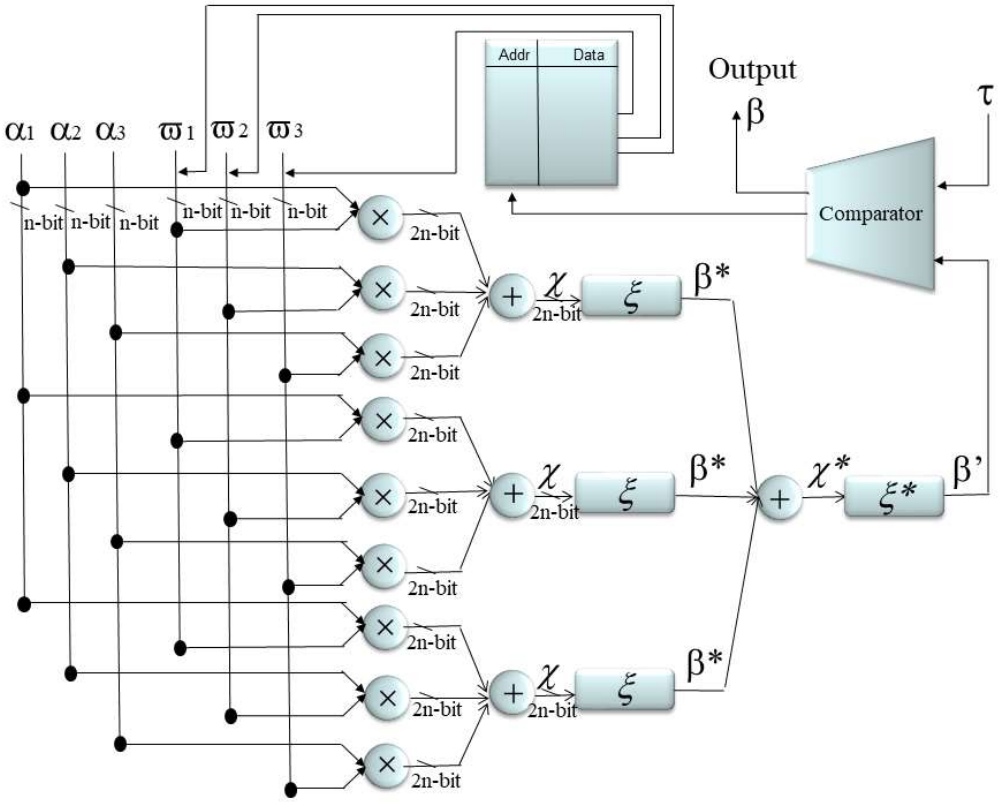

Figure 5. Diagram of the hardware-based the backward ANN.

The design of the backward ANNs can add the comparator at the output β' to compare the value of output β' with the desired value τ. If it is granted so the result will be obtained but when it is not so the weights will be updated using the value that is taken from the LUT. The LUT in here can be designed as a RAM or ROM.

5. Conclusion

This paper has described the platform of the hardware-based the algorithm of the artificial neural network. The hardware platform, in this case, has used FPGAs as the parallel processing. The multiplication and the adder are very easy to do to obtain the output of the multiplication between input and weights. One of the difficulties in the ANN implementation is to design the activation function because it has many variations. The future work for this implementation is to analyze the design of the activation function with some design variations.

Acknowledgments

The author wishing to thank for the assistance and encouragement from the Hemispheres team who has given the author chance to research and present this paper.

6. References

[1] Wibowo FW Sudarmawan and Sulistiyono M 2018 Hardware platform design analysis of K-Means clustering algorithm implementation *International Journal of Engineering and Technology(UAE)* vol 7 Issue 4 pp 90-93 DOI: 10.14419/ijet.v7i4.40.24082

[2] Altera 2008 FPGA run-time reconfiguration: two approaches *White Paper* ver 1.0

[3] Wibowo FW 2011 Interoperability of reconfiguring system on FPGA using a design entry of hardware description language *2011 Computation and Communication Technologies: 3rd International Conference on Advances in Computing, Control, and Telecommunication Technologies*, *ACT 2011 - Computer Science Series* 1 pp 79-83.

[4] Dondon P Carvalho J Gardere R Lahalle P Tsenov G Mladenov VM 2014 Implementation of a feed-forward artificial neural network in VHDL on FPGA *Proceedings of the 12th Symposium on Neural Network Applications in Electrical Engineering NEUREL* pp 37-40 DOI: 10.1109/NEUREL.2014.7011454

[5] Hariprasath S Prabakar TN 2012 FPGA implementation of multilayer feed forward neural network architecture using VHDL *Proceedings of 2012 International Conference on Computing, Communication and Applications* pp 1-6

[6] Dorafshan N 2012 FPGA implementation of neural networks *Presentation in Semnan University – Spring*

[7] Braga ALS Llanos CH Ayala-Rincon M Jacobi R 2005 VANNGen: a flexible CAD tool for hardware implementation of artificial neural networks *Proceedings of ReConFig 2005: 2005 International Conference on Reconfigurable Computing and FPGAs* pp 8-13 DOI: 10.1109/RECONFIG.2005.35

[8] Wibowo FW 2018 Comparison of multiplication algorithms based on FPGA *Proceedings of 2018 2nd Borneo International Conference on Applied Mathematics and Engineering (BICAME)* pp 326-331

[9] Wibowo FW 2018 Implementation of FPGA in index data storage as a database *International Journal of Engineering and Technology(UAE)* vol 7 Issue 4 pp 94-97 DOI: 10.14419/ijet.v7i4.40.24083