EMITTER International Journal of Engineering Technology
Vol. 7, No. 1, June 2019
ISSN: 2443-1168

FPGA Based Design of Artificial Neural Processor Used for Wireless Sensor Network

Azzad Bader Saeed, and Sabah Abdul-Hassan Gitaffa

Electrical Engineering Department, University of Technology, Baghdad, Iraq.
30065@uotechnology.edu.iq, 30094@uotechnology.edu.iq

Abstract:

In this paper, a simulation of artificial intelligent system has been designed for processing the incoming data of sensor units within Wireless Sensor Network WSN, and then presenting proper decision. The Back-propagation Neural Network BPNN has been used as the proposed intelligent system for this work, whereas the BPNN is considered as a trained network in conjunction with an optimization method for changing the weights and biases of the overall network. The main two features of the BPNN are: high speed processing, and producing lowest Mean-Square-Error MSE (cost function) in few iterations. The proposed BPNN has used the linear activation functions 'Satlins' and 'Satline' for the hidden and output layer respectively, and has used the training function 'Traingda' (which is gradient descent with adaptive learning rate) as a powerful learning method. It is worth to mention, that no previous research used these three functions together for such analysis. The proposed system has been designed, and tested using MATLAB software package. An optimal result has been obtained in this work, where the Mean-Square-Error MSE has reached to 'zero' value in 87 iterations, which leads to the fitting between real and desired outputs. In fact, there is no previous work has reached this optimal result yet. The proposed BPNN has been implemented in FPGA (Field Programmable Gate Array), which is fast and low power tool. After implementing the proposed intelligent system in the FPGA, the maximum processing time (i.e. proposed system delay) has reached to 100nsec.

Keywords: Artificial Intelligent System, Back-propagation Neural Network BPNN, FPGA, Mean-Square-Error MSE, Wireless Sensor Network WSN.

1. INTRODUCTION

The most used application today is the Intelligent Wireless Sensor Network, which consists of three main parts: the sensor nodes, control station (HUB), and the Artificial Intelligent System AIS. The sensor node is constructed from the sensor or sensors, and Analog-to-Digital converter, the sensor is used to translate the physical event to analog electrical signal, and the Analog-to-Digital converter is used to change this analog electrical signal to digital form (Binary). The sensor node may consists of a local processing unit is used for initial processing the data of the sensor node before going to the control station. The sensor nodes is connected to the control station by a wireless communication system through a channel. The control station is a Central Processing Unit for the overall network is used to process the overall data incoming from all of sensor nodes. The Artificial Intelligent System is used as a processing network for the incoming data, which is essentially used in the control station, or it may be used in the local processing of the sensor nodes. In fact, the intelligent system is a software, which can be executed by a computer system, or it can be implemented in FPGA (Field Programmable Gate Array), as known the FPGA is a hardware is characterized by its: fast processing, low power consumption, easy of installation [1][2][3].

Usually, the Back-propagation Neural Network is used as an Artificial Intelligent System for the wireless sensor networks, which is considered as an effective network, it has best learning method. The Back-propagation Neural Network BPNN is training algorithm and an optimization process, the training algorithm is used for learning the network by entering pattern of data, while the updating the weights and biases are performed by the optimization process. In the Back-propagation process , the gradient descent expression is depended, the gradient descent is defined as the derivative of the loss function to the weights and biases of the overall network [4][5].

A target or desired output is required for algorithm of the Back-propagation Neural Network, which is used for the loss function of the gradient descent. The Back-propagation is considered as a multi-layer feed-forward network with delta role for learning process. The Back-propagation Neural Network usually involves three or more layers; single input layer, single or multi- hidden layer, and single output layer. The BPNN needs activation functions in the hidden and output layers, which they are used for processing the incoming data to these layers, also the BPNN needs a training function for its learning process. In each iteration, the BPNN algorithm calculate the gradient descent for hidden and output layers [6][7][8].

The process of the Back-propagation starts with entering the input data pattern to the input layer, then these data propagate to the hidden layer, at this time the output of the hidden layer will be generated using the activation function of this layer, then the output data propagate to the output layer, in this time the output data will be generated at the output of the output layer, the a loss function will be presented, which is the difference between the real output of the output layer and the output target, then this loss function is used to change the weight and biases of the overall network [9][10].

Suppose the error e_o for a specific output unit (o). The error function e_o must reach to zero value for optimal training, which is performed by changing the weights of the connection of the network to suitable values. In delta rule, the error function can be reduced by adapting the incoming weights according to the following expression [11]:

$$w_{ho} = (d_o \quad y_o)\, y_h \tag{1}$$

where,
Δw_{ho} : Changing of connections weights between output and hidden layers.
d_o : Desired (Target) output of output layer..
y_o : Real output of output layer .
y_h : Real output of hidden layer.

This weights adaption for the output layer. For adapting the weights from input to hidden units, the delta rule must be utilized again. At the beginning of this step, the delta function δ of the hidden units hasn't a value. The chain rule is used to solve this problem, which is performed by distributing the error of an output unit (o) to each hidden unit. In other word, a delta function can be received by the hidden unit from the corresponding output unit (o) equals to the delta function of corresponding output unit multiplied by the connection weight between the corresponding hidden and output units.

The algorithm of the general delta rule is performed in two steps. At first step, the input data pattern (x) enters to the input layer , and then propagates in forward direction through the hidden and output layers to generate the real output (y_o^p) for every output neuron, then this output and its desired (Target) value d_o will be compared to generate the error signal (δ_o^p) for each output neuron. At second step, this error signal passes through the network in backward direction to generate the new proper weight for each neuron in the hidden and output layers of the network. Finally, the updating to all the weights of the network will be performed.

The connection weight will be updated by a specific value, this value is proportional to the production of the error signal δ of the unit k (receiving the input) and the output of the unit j (sending this signal along the connection), which can expressed as follows [11]:

$${}_p w_{jk} = \gamma \delta_k^p y_j^p \tag{2}$$

where,
Δw_{jk} : Changing in weight to the pth input connection unit between (j)
Sending Layer and (k) receiving layer.
γ : Rate of Learning.
δ_k^p :Delta function for pth neuron of the (k) receiving layer.
y_j^p : Real output of pth neuron of the (j) sending layer.

For the output neuron, the error signal can be expressed as follows:

$$\delta_o^p = \left(d_o^p \quad y_o^p \right) F'(s_o^p) \tag{3}$$

where,

δ_o^p : Delta function of pth neuron of the output layer.
d_o^p : Desired output of pth neuron of the output layer.
y_o^p : Real output of pth neuron of the output layer.
$F'(s_o^p)$: Differentiation of activation function of pth neuron of the output layer.

If the activation function F is 'sigmoid' , then it can be expressed as follows:

$$y^p = F(s^p) = \frac{1}{1+e^{-s^p}} \tag{4}$$

and the derivative of this function can be given as follows:

$$F'(s^p) = y^p(1 \quad y^p) \tag{5}$$

The error signal of an output can be written as follows [12]:

$$\delta_o^p = \left(d_o^p \quad y_o^p\right) y_o^p \left(1 \quad y_o^p\right) \tag{6}$$

The error signal of a hidden neuron can be calculated using the error signal of the output neuron that is directly connected to it and its weight connection, which is given by the following expression [11]:

$$\delta_h^p = F'\left(s_h^p\right) \sum_{o=1}^{N_o} \delta_o^p w_{ho} = y_h^p \left(1 \quad y_h^p\right) \sum_{o=1}^{N_o} \delta_o^p w_{ho} \tag{7}$$

where,

δ_h^p : Delta function of pth neuron of the hidden layer.
$F'(s_h^p)$: Differentiation of activation function of pth neuron of the hidden layer.
w_h^o : Weight connection between hidden and output layer.
y_h^p : Real output of pth neuron of the hidden layer.

Note, equation 7 has been concluded for sigmoid activation function.

In this work, a simulation of Back-propagation neural network has been designed, and implemented in FPGA (type: Xilinx Spartan 6 SP-605 evaluation kit). This simulation software has been used for presenting the average value of incoming data of two sensor nodes within Carbon Monoxide Wireless Sensor Network (COWSN). Each sensor node produces 2-bits data for the proposed simulation system that is implemented in the FPGA. This 2-bits data can represents three possible states, where, the binary data (01) represents the NORMAL (LOW) state, the binary data (10) represents the CRITICAL (MEDIUM) state, and the binary data (11) represents FATAL (HIGH) state. The proposed simulation system has three output lines (z,y,x), which it can represents three possible data, the output data (0,0,1) represents NORMAL (LOW) state, the output data (0,1,0) represents the CRITICAL (MEDIUM) state, and the output data (1,0,0) represents FATAL (HIGH) state.

The proposed simulation system consists of three layers: Input layer, single hidden layer, and output layer. The input layer contains 4 neurons where, each two neurons relates to output data of one of the two sensor node, the hidden layer involves 10 neurons, and the output layer involves 3 neurons. In this work, accurate results has been obtained due to the following reasons: 1). 'Traingda' MATLAB function has been used as training function for the proposed neural network, 2) 'Satlins' MATLAB function has been used as linear activation function for the hidden layer, 3) 'Satlin' MATLAB function has been used as linear activation function for the output layer. Due to using 'linear' activation functions for the proposed intelligent system, this system has been easily converted to VHDL ((Very high-Speed Integrated Circuit) Hardware Design Language) code program, and then has been uploaded in FPGA successfully, whereas, if non-linear activation functions are used, the simulation system cannot be converted to VHDL code program and then cannot be uploaded in FPGA.

There are two reasons for using the Back-propagation neural network as an intelligent system in this work, they are: 1) The fast processing of this neural network due to the calculation of the gradient descent in its algorithm at each training iteration, the gradient descent is the partial derivative of the error function with respect to any weight and bias of the network, it represents the fast changing in the error function due to the changing of the weights and biases at each iteration, i.e. the gradient descent starts with maximum value at first iteration, and then it quickly decreases to minimum value in few iterations. 2) The high accuracy training of this neural network, whereas, this network generates two error functions, the first is generated from the output layer, which is used to update the weights and biases of input connections the output layer, while the second is generated from the hidden and output layers, which is used to update the weights and biases of input connections of the hidden layer, so, the Back-propagation neural network can present minimum error function in few iterations [4][11].

The proposed simulation system has utilized the FPGA for implementing its network due to the following reasons:1) Fast processing of the FPGA because of its parallel processing, 2) Low power consumption, 3) Easy of implementing any network in FPGA.

The MATLAB software package has been used to built the proposed simulation system, and the ISE Design Suit software package has been used to upload the proposed VHDL code program in FPGA.

2. RELATED WORKS

K. Henkel and D. Schmeiber, 2002 [12], in this work, the problem of independently estimating of the carbon dioxide concentration and humidity of a gas mixture had been solved, where the signals of two quartz microbalances coated by sensitive polymer layer has been recorded. Two Back-propagation neural networks had been proposed in this work, the first is single-stage network, and the second is two-stage network (which involves two single-stage networks connected in series) with respect to their generalization ability. The proposed networks of this work is not implemented in FPGA due to the using the non-linear activation functions for the hidden and output layers.

Muhammed K., B. Muhammed K. and T. Muhammed J., 2012 [13], in this work the Back-propagation neural network had been used for an application of wireless sensor network. This work had been performed using FPGA and MATLAB software package. In this work the environmental conditions such as temperature and humidity had been studied and realized using neural network and wireless sensor network in a base station. Analog-to-Digital converter ICs and microcontrollers has been utilized in the sensor units of the proposed sensor network.

Guo W. and Juan W., 2014 [14], in this work, the nodes of wireless sensor network had been designed and implemented, which has been based on Back-propagation neural network. The proposed neural network is trained by specific parameters of protocol of routing for mobile type of wireless sensor networks, which may be active or passive depends on the proposed system design. This proposed neural network of this work is not implemented in FPGA due to the using the non-linear activation function for the hidden and output layers.

Azzad B. Saeed, 2018 [15], has designed a simulation of Back-propagation neural network BPNN as controlling intelligent system for automotive transmission gearbox of automobiles, whereas, this neural network has been used for automatic controlling on proper selecting of gear ratios for the transmission gearbox . In this work, The MATLAB tools 'Satlins' and 'Satlin' had been used as linear activation function for his proposed design, while, the tool 'trainlm' (Levenberg-Marquardt training function) has been used as a training function for the same proposed network. The proposed neural network of this work has been implemented in FPGA (type Xilinx Spartan6 SP-605) successfully.

3. PROPOSED METHOD

The related Wireless Sensor Network WSN for the proposed work is used for sensing and determining the average level of Carbon Monoxide Gas over a limited indoor area such as industry, or factory, ...etc. This WSN consist of three main parts:1) Two sensor nodes, whereas, each sensor node is used for sensing the CO gas and converting the level of this gas to 2-bits binary data. The 2-bits data has been chosen in the proposed design due to following reasons, a) in fact, the accuracy of gas concentration is not interested, but only the fundamental levels of the CO gas is considered, Normal, Critical, and Fatal(Low, Medium, and High), so, three levels of this gas has been considered in this design, which is covered by representing them by 2-bits data, b) to increase the data rate (speed of data) as more as possible. 2) Laser communication system, which is used to transport the CO level data and the key signals between the sensor nodes and the control station, The laser technique is utilized for this work due to its excellent characteristics such as low consumption power, ultra wide bandwidth, and fastest data transporting. 3) The control station (HUB), which is the proposed intelligent system (i.e. Proposed neural network) that is implemented in the FPGA, the role of this part is to present: a) the average level of the CO gas according to the output data of the two sensor nodes, b) the key signals for turning the sensor nodes ON and OFF at specific times.

The Back-propagation neural network has been used as an intelligent system in this work, and the MATLAB software package is used for writing the proposed program and generating the proposed simulation block. The proposed design has been performed with the following requirements: 1) Two sensor units is consisted in the related sensor network, 2) Two output bits are chosen for every sensor unit, they are connected to the input lines of the proposed simulation block, therefore, the proposed simulation has four input lines, each two input lines are connected to the output of sensor unit. 3) Three possible output data are presented at the output of each sensor unit, they are, the binary data (01) which corresponds to LOW state, the data (10) corresponds to MEDIUM state, and the data (11) corresponds to HIGH state. 4) Three lines are chosen for the output of the proposed simulation block. Three output states are presented at the output of the proposed simulation block, they are, the LOW state is represented by the binary data (001), the MEDIUM state which is represented by the binary data (010), and the HIGH state is represented by the binary data (100). For this reason, the proposed simulation block four input lines and three output lines. 5) The proposed system has been designed to present the average value of the outputs of the two sensor units, the relation between input and desired output data is illustrated in Table 1. The output data of the sensor unit (1) are connected to the input lines of the proposed system 'a' and 'b', 'a' represents the least significant bit, while 'b' represents the most significant bit. The output data of the sensor unit (2) are connected to the input lines of the proposed system 'c' and 'd', 'c' represents the least significant bit, while 'd' represents the most significant bit. The desired output data of the proposed simulation block are 'x', 'y', and 'z'. As shown in Table 1, if the incoming data from output of the sensor unit is (ab=10) which represents LOW state, and the incoming data from output of the sensor unit 2 is (cd=10) is also represents LOW state, then the desired output data of the proposed system is (xyz=100) which represents LOW state, and so on. The proposed neural network (or proposed system) has been designed by using: a) four neurons in the input layer which correspond the four input lines, b) three neurons in the output layer which corresponds to three output lines of the proposed system, c) ten neurons in the single hidden layer, which has been chosen for obtaining optimal result.

Table 1. The relation of input data with desired output data of the proposed system.

Input Data				Desired Output Data		
From Sensor 1		From Sensor 2				
a	b	c	d	x	y	z
1	0	1	0	1	0	0
0	1	1	0	0	1	0
1	0	0	1	0	1	0
0	1	0	1	0	1	0
1	0	1	1	0	1	0
1	1	1	0	0	1	0
1	1	0	1	0	0	1
0	1	1	1	0	0	1
1	1	1	1	0	0	1

The proposed system has utilized the two linear activation functions 'Satlins' and 'Satlin' (which are shown in Figure 1) for the hidden and output layers respectively, these two function are shown in Figure 1. The proposed system has used the MATLAB training 'Traingda' (it is training of gradient descent with adaptive learning rate), this learning function has been used for changing the weights and biases of input connection of the proposed system to suitable values for obtaining accurate results. Also for obtaining accurate results, the learning rate has been set to 0.05, and the maximum training iteration has been set to 300.

The process of building and training the proposed neural network is illustrated in the flowchart in Figure 2, as shown in this flowchart, the building of the proposed network starts from setting the neurons of the input, hidden, and output layers, then setting the activation functions for the hidden and output layer, then setting the training function, then setting the learning rate and maximum no. epochs. The learning process starts from entering the first input data pattern, at same time, the desired output must be set, then the Back-propagation process will start, then it finished with generation the loss function that is used for updating the weights and biases of the network. The second training process will start with entering the next input data pattern, and setting the new related desired output data, and so on.

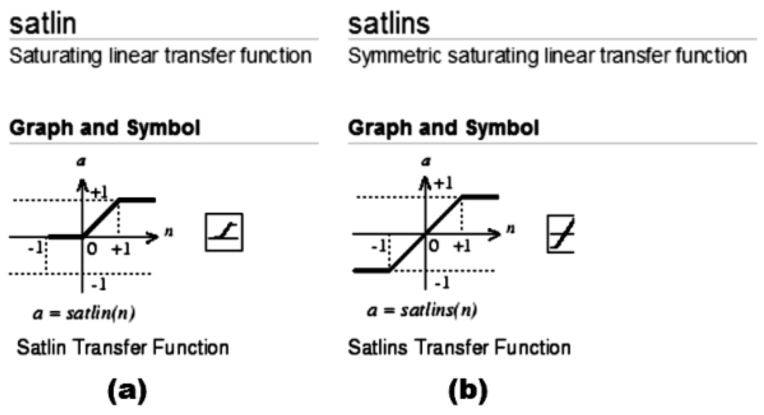


Fig. 1 Linear activation functions[13]: a) Satlin function. b) Satlins function.

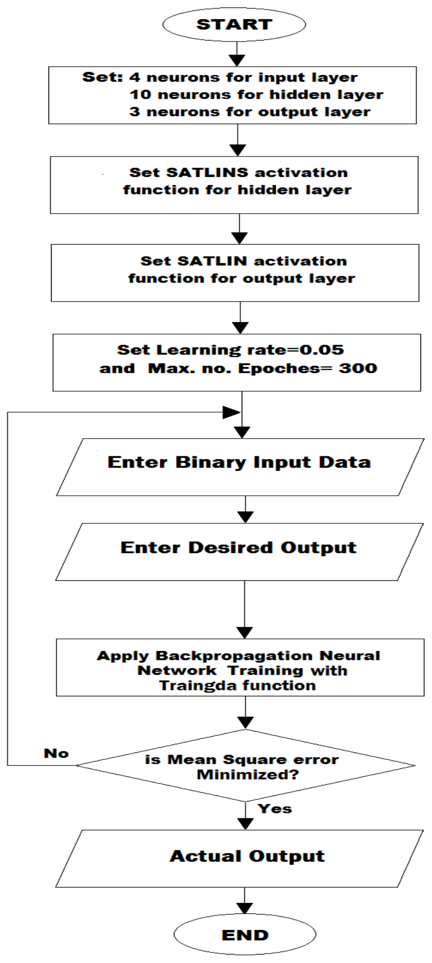

START
Set: 4 neurons for input layer
10 neurons for hidden layer
3 neurons for output layer
Set SATLINS activation function for hidden layer
Set SATLIN activation function for output layer
Set Learning rate=0.05 and Max. no. Epoches= 300
Enter Binary Input Data
Enter Desired Output
Apply Backpropagation Neural Network Training with Traingda function
is Mean Square error Minimized?
No
Yes
Actual Output
END

Fig. 2 Flowchart of the proposed system learning.

The proposed program of Back-propagation neural network has been written using MATLAB software package, which is ended by the statement {gensim(net)}. A proposed simulation block will present on the screen of the computer after execution of the proposed program. This simulation block has four input lines that is connected to the output of the sensor units, and has three output lines which is connected to LEDs (light Emitting Diodes). This simulation block deals with logic (digital) data.

By clicking on 'Look Under Mask' option, another network will present, which is shown in Figure 3, this network has: 1) four input ports, which is used to enter the input data, 2) a multiplexer, which is used to convert the parallel input data to serial data, 3) ' data type conversion 1', which is used to convert the data type from 'logic' to 'fixdt' (fixed-point or floating point data), 4) 'Layer 1' block, which represents the input and hidden layer of the proposed system, 5) 'Layer 2' , which represents the output layer of the proposed system, 60 another ' Data Type Conversion', which is used to convert the data type from 'fixda' to 'logic', 7) A de-multiplexer, which is used to convert the serial output data to parallel.

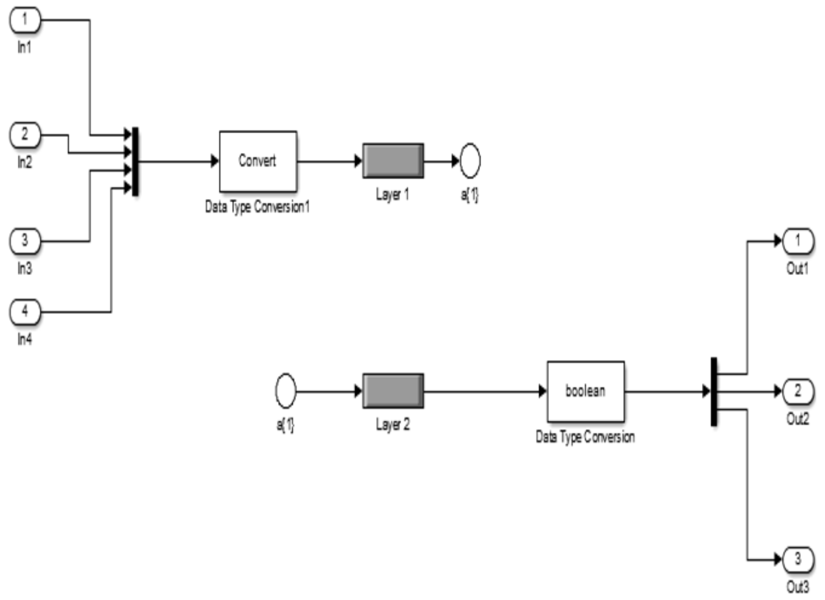


Fig. 3 Internal composition of the proposed simulation block.

The composition of weight block of the 'Layer 1' block is shown in Figure 4, as shown, this weight block involves ten units, each unit represents the production of a weight and the corresponding input data incoming from the input layer, these units represent the neurons of the hidden layer of the proposed simulation system.

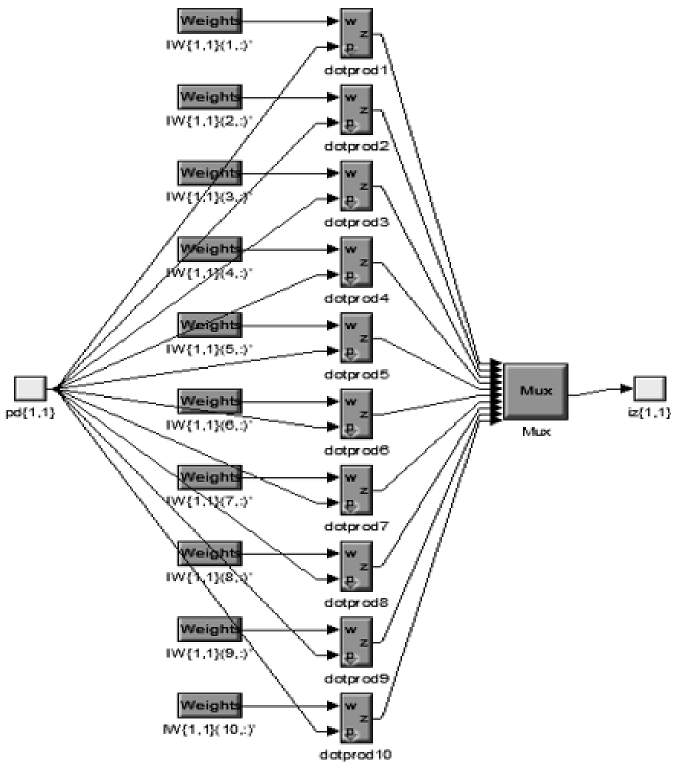


Fig. 4 composition of the weight block of 'layer 1' block.

The composition of weight block of the 'Layer 2' block is shown in Figure 5, as shown, this weight block involves three units, each unit represents the production of a weight and the corresponding input data incoming from the hidden layer, these units represent the neurons of the output layer of the proposed simulation system. After clicking on the performance option, a curve line will be exhibited, as shown in Figure (6) , one can see from this figure that the mean square error gradually decreases in continuous manner to reaches the zero value, which leads to the successful learning with time proceeding.

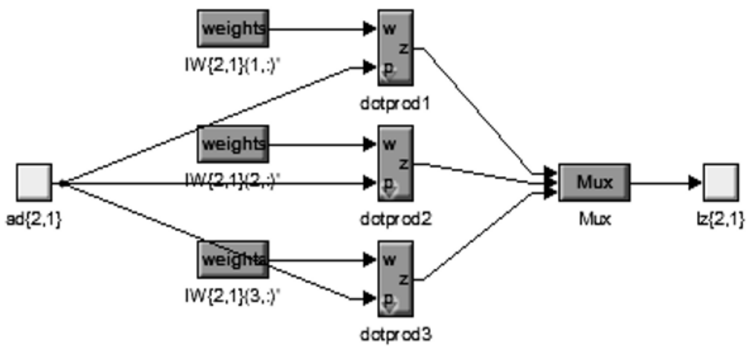


Fig. 5 Composition of weight block of 'layer 2' block

The block diagram of the proposed software system with the related hardware system is illustrated in Figure 6. As shown in this figure, the overall process is performed by the following steps: 1) writing a suitable MATLAB program for building the proposed Back-propagation neural network, 2) then this proposed MATLAB program is executed to generate a simulation block system on the screen of the computer, 3) the generated simulation block system is then converted to VHDL code program using MATLAB software package, 4) the final VHDL code program is then uploaded to the FPGA through JTAG (Joint Test Action Group)serial input bus, i.e. in this step the proposed simulation system will be implemented in FPGA. The related hardware system (Carbon Monoxide Wireless Sensor Network COWSN) involve three main units: 1) The sensor nodes, 2) Laser communication system, 3) The FPGA which acts as a HUB or control station used for processing the data incoming from the two CO sensor nodes. As shown from the block diagram of Figure 6, the FPGA has been designed to have 4-useful input lines, and 3-useful output lines. The 4-input lines is connected to the output of the two sensor nodes, where each 2-input lines is connected to related output of one of the two sensor nodes, note each sensor node can deliver 2-bits data according to the sensed level of the carbon monoxide gas, which sensed by the sensor. The 3-output lines of the FPGA is used to present the output state of the proposed intelligent system according to the incoming data of the two sensor nodes.

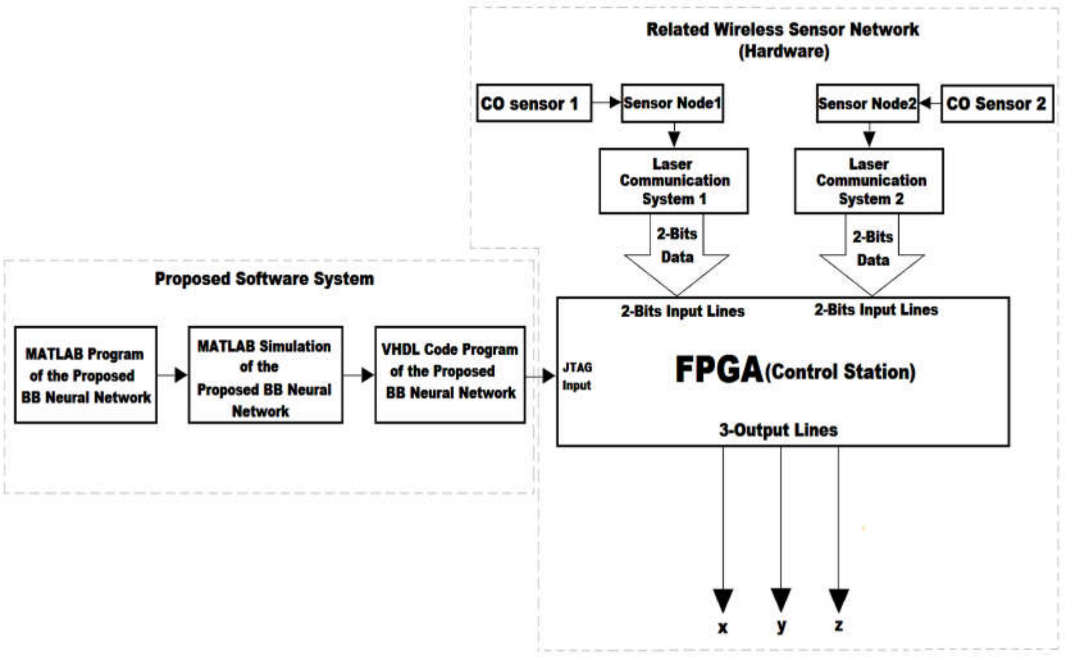

Related Wireless Sensor Network (Hardware)
CO sensor 1
Sensor Node1
Sensor Node2
CO Sensor 2
Laser Communication System 1
Laser Communication System 2
2-Bits Data
2-Bits Data
Proposed Software System
2-Bits Input Lines
2-Bits Input Lines
MATLAB Program of the Proposed BB Neural Network
MATLAB Simulation of the Proposed BB Neural Network
VHDL Code Program of the Proposed BB Neural Network
JTAG Input
FPGA(Control Station)
3-Output Lines
x
y
z

Fig. 6 The block diagram of the proposed software system with the related hardware system.

4. EXPERIMENT AND ANALYSIS

After testing the proposed Bp neural network, a performance result has been presented on the screen of the computer, which is illustrated in Figure 6, it represents the relation between the Mean-Square-Error MSE and no. iterations (Epochs). As shown in Figure 7, the MSE value begins from nearly 1 at epoch 0, then it gradually decreased to zero at epoch 87, which is considered an optimal result, because when the MSE reaches to zero value then the desired output data is fitted to the real output data.

The regression result of the proposed simulation system is illustrated in Figure 8, the regression result represents the relation between the real output data and the desired output data. In Figure 8, there are two fitted relation lines, the first one is solid line which represents the fitting relation between the real and desired outputs, and the second one is dotted line which represents the actual relation between the real and desired outputs. The fitting between these relation lines means that the real output is fitted to the desired output from the starting to finishing the training process.

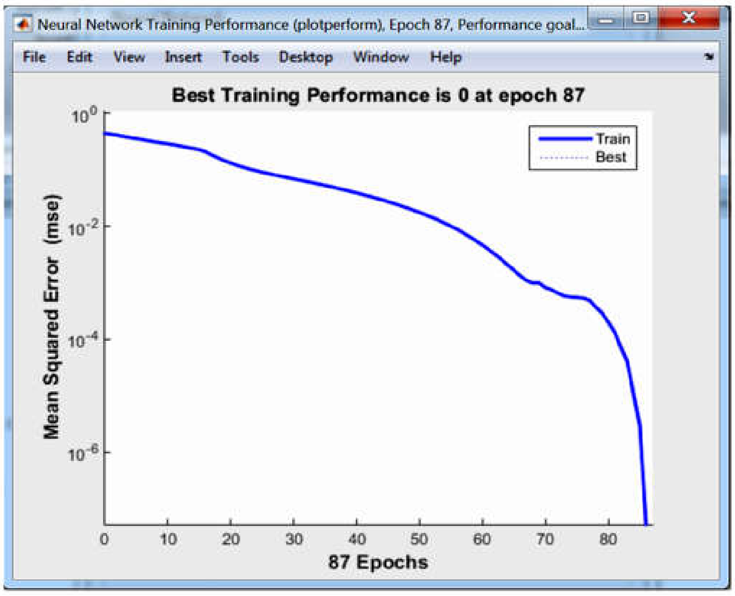


Fig. 7 Performance result of the proposed simulation system.

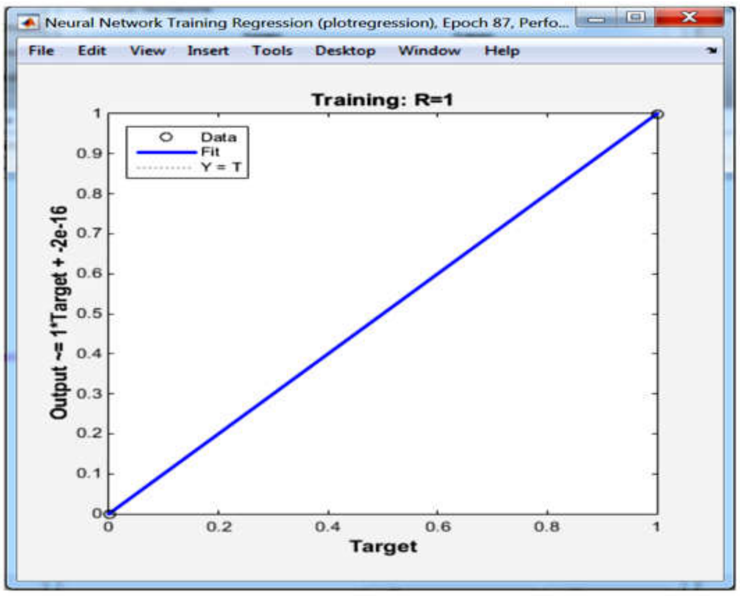


Fig. 8 Regression output of the proposed simulation system.

The behavior of the proposed simulation system is demonstrated in the state transition diagram that is shown in Figure 9, it presents the state transitions of the three outputs of the proposed system (x,y,z), which is related to the state transitions of the four input (a,b,c,d) of the same system. The state diagram of the proposed system.

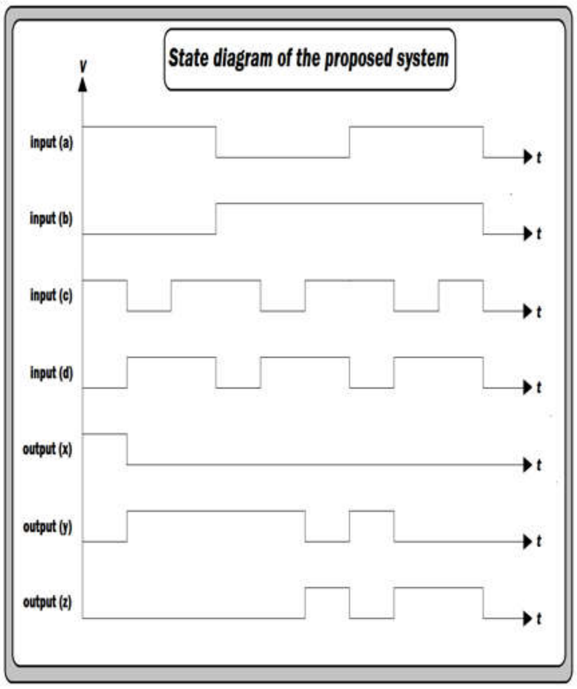


Fig. 9 State diagram of the proposed Simulation system

K. Henkel and D. Schmeiber, 2002 [12] have realized a Back-propagation neural network for a sensor network. In this work, non-linear activation functions are utilized for the hidden and output layers, and the training function 'Traingd' (Gradient Descent) is utilized as a learning method for their proposed work. They had proposed two networks: a single-stage network, and a two-stage network. Figures 10 and 11 illustrate the performance results of the single-stage and two-stage networks for this work. Figure 9 shows that the MSE value for the single-stage had reached to 0.118 at epoch 300, while Figure 10 shows that the MSE value for the dual-stage network had reached to 0.014 at epoch 300. One can observe the MSE results of this work has so higher value than the MSE result of the proposed work in this paper that is illustrated in Figure 7. Due to using non-linear activation functions in this work, the simulation system of this work has been implemented in FPGA.

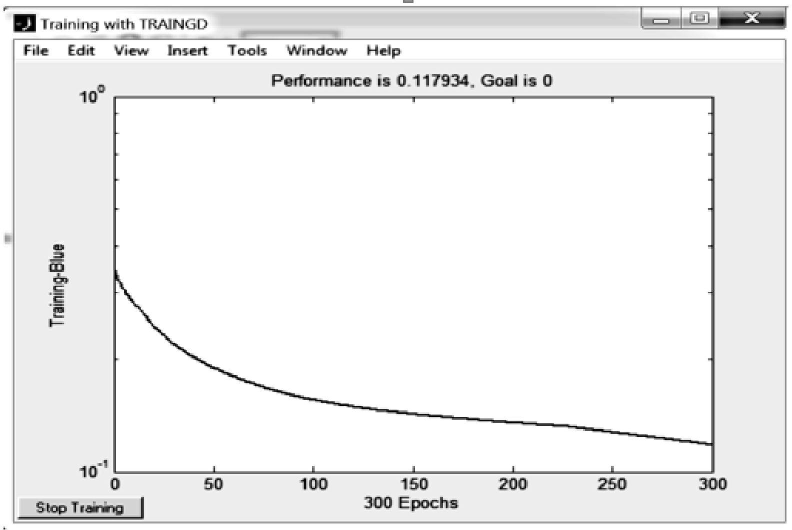


Fig. 10 Performance result of single-stage Bp neural network of the related work [12].

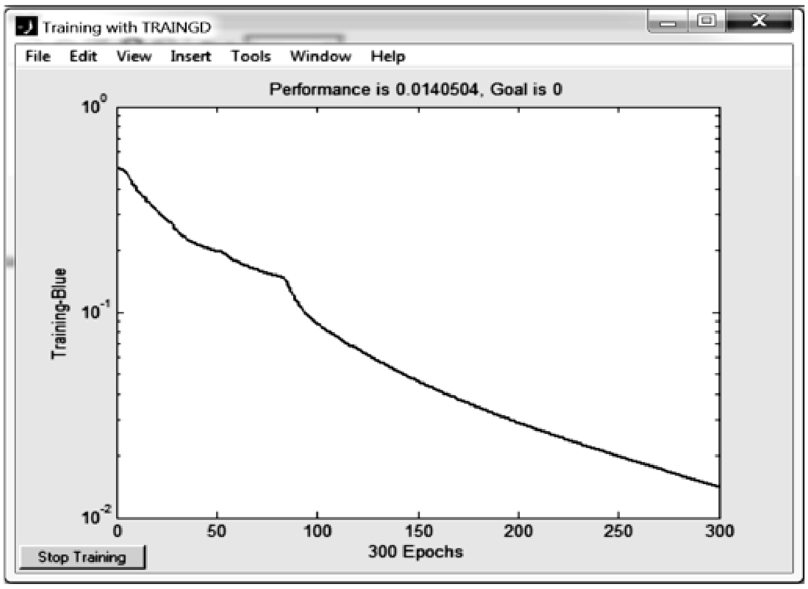


Fig. 11 Performance result of the two-stage Bp neural network of the related work

Muhammed K. and B. Muhammed, 2012 [13] had realized a Back-propagation neural network for wireless sensor network. The training function 'Trainlm' (Levenberg-Marquardt) had been used as a learning method for this work. In this work, linear activation functions had been used for the hidden and output layers respectively. Figure 12 presents the performance result of this work, as shown in this figure, the MSE value has finished to 3.53527×10^{-30} in 16 iterations, which is acceptable value, but it still more than that of the proposed work of this paper that is shown in Figure 7. Due to using the linear activation function in this work, this proposed system of this work has been implemented in FPGA.

Guo W. and Juan W., 2014 [14] had achieved a Back-propagation neural network for a sensor network. Non-linear activation functions had been used for hidden and output layers of this work. The training function 'Traingdx' (Gradient Descent with Momentum and Adaptive Learning Rate) had been utilized as learning method for the network of this work. Figure 13 presents the performance result of this work, as shown in this figure, the MSE value had ended to 4.415×10^{-12}) in 90 epochs, it is also an acceptable value, but it still so higher than that of the proposed system of this paper that is illustrated in Figure 7. The proposed simulation of this work had not been implemented in FPGA due to using non-linear activation function in the hidden and output layers.

Azzad B. Saeed, 2018 [15] has designed a controlling neural network involves three main layers: 1)The input layer which consists of eight neurons, 2) The single hidden layer which consists of five neurons, 3) The output layer which also consists of five neurons. The utilized activation functions for this work are 'Satlins' and 'Satlin', while the utilized training function is 'Trainlm'. The performance result of this work is illustrated in

Figure 14, as shown from this figure, the curve of MSE value starts from nearly 1 at epoch 0, and has gradually finished to 3.94×10^{-25} at epoch 15, which is also acceptable value, but it is still more than that of the proposed work of this article that is shown in Figure 7.The training process of this work has been performed in 15 iterations, which is faster than that of the work of this article (87 iterations).

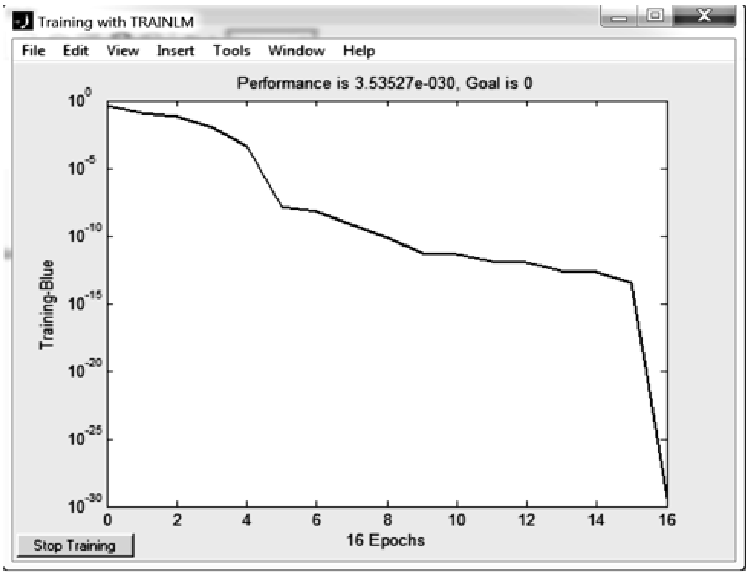


Fig. 12 Performance result of the Bp neural network of related work [13].

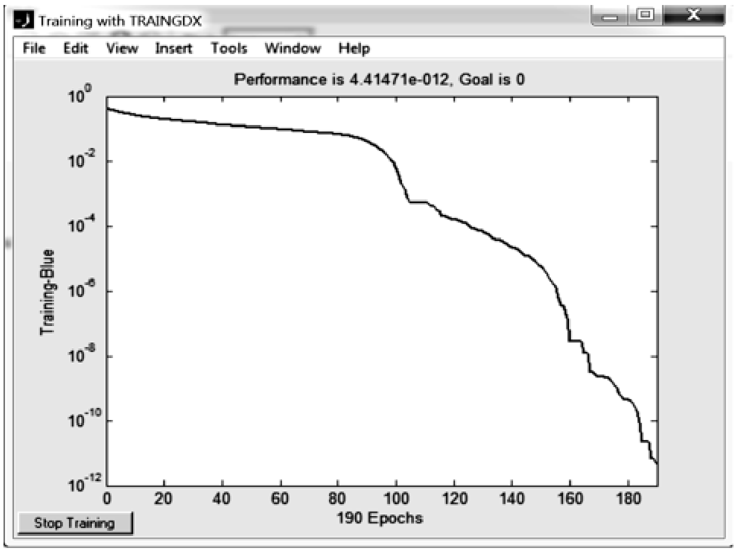


Fig. 13 Performance result of the Bp neural network of related work [14].

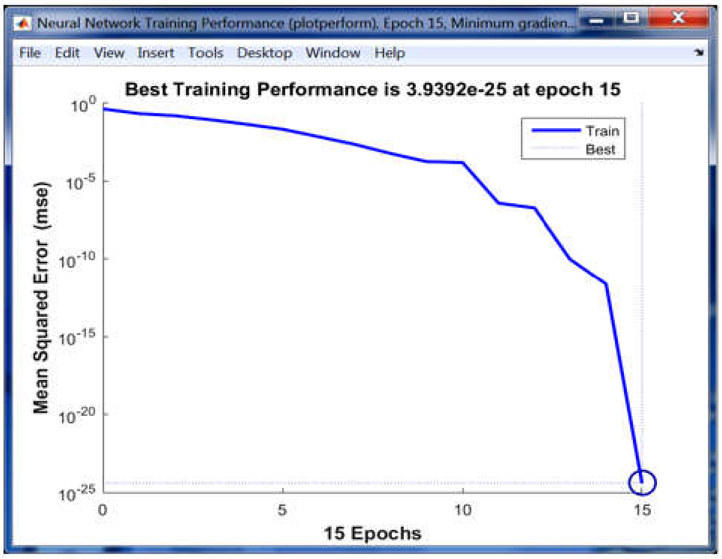
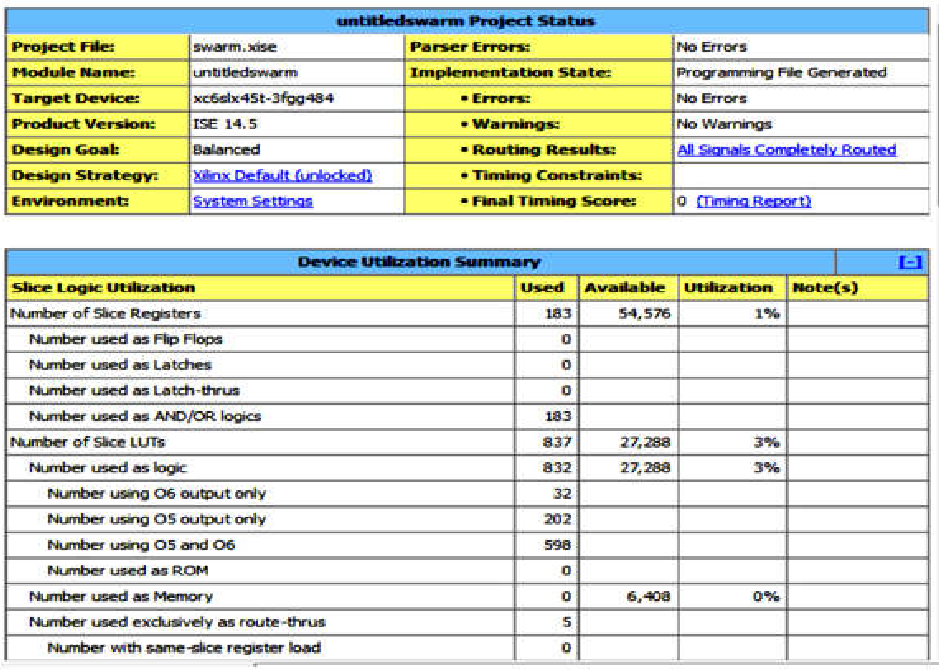


Fig. 14 Performance result of the proposed BP neural network of related work [15].

The proposed Back-propagation neural network has been implemented successfully in FPGA (type: Xilinx SPARTAN 6 SP605 Evaluation Kit), which is performed using ISE Design Suit software package. The two pages of list of used logic resources of the FPGA are presented in Figures 15 and 16, as shown from these figures, no. used slice registers is 183, no. used slice Look-Up-Tables (LUTs) is 837, no. used AND/OR gates is 183, no. occupied slices is 278, no. used Digital Signal Processors (DSPs) is 48, and no. Input-Output Blocks (IOBs) is 8, and so on. The ultimate implementation of the related CO wireless sensor network which is connected to the FPGA kit is illustrated in Figure 17.

After implementing the proposed Back-propagation neural network in the FPGA, the execution time of the overall system has reached to 100 nsec, which is so slight and acceptable value. This execution time has been obtained due to the using of the Back-propagation neural network (which is fast intelligent system), and the FPGA (which is fast hardware tool).

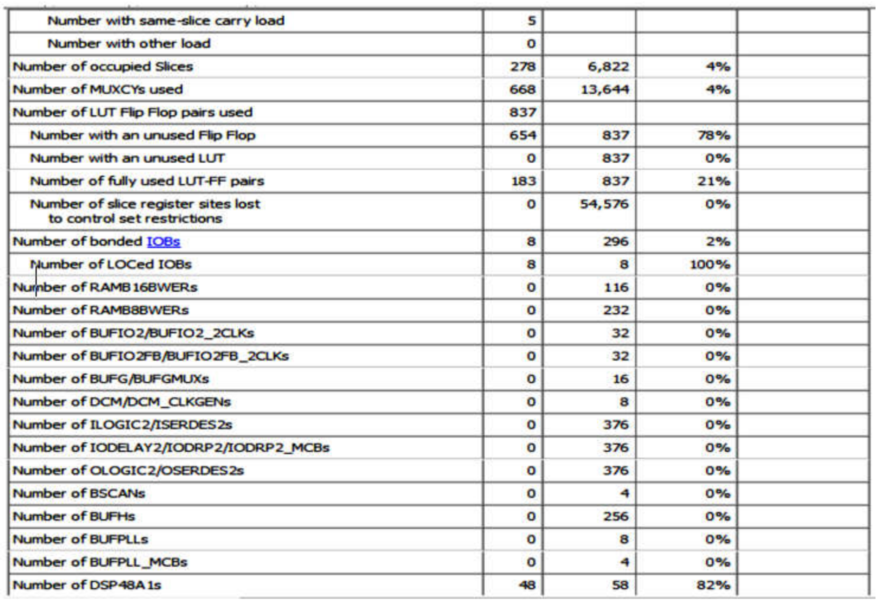
There are four main characteristics must be considered for implementing the proposed simulation system in FPGA: the speed (bit rate), consumption power, size, and capability of upgrading. In this work, the proposed system has been implemented using intelligent simulation system based on FPGA, as known, the FPGA is characterized by its low power consumption, high speed (high bit rate reaches to 200 Mbits/sec), small size of implementing, and the capability of upgrading. The low power consumption and high speed of processing is due to the using of FPGA which is hardware), and the capability of upgrading is due to the using of intelligent simulation system (which is software).



untitledswarm Project Status			
Project File:	swarm.xise	**Parser Errors:**	No Errors
Module Name:	untitledswarm	**Implementation State:**	Programming File Generated
Target Device:	xc6slx45t-3fgg484	**• Errors:**	No Errors
Product Version:	ISE 14.5	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	183	54,576	1%	
Number used as Flip Flops	0			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	183			
Number of Slice LUTs	837	27,288	3%	
Number used as logic	832	27,288	3%	
Number using O6 output only	32			
Number using O5 output only	202			
Number using O5 and O6	598			
Number used as ROM	0			
Number used as Memory	0	6,408	0%	
Number used exclusively as route-thrus	5			
Number with same-slice register load	0			

Fig. 15 First report of final results for implementing the proposed neural network in the FPGA.



Number with same-slice carry load	5			
Number with other load	0			
Number of occupied Slices	278	6,822	4%	
Number of MUXCYs used	668	13,644	4%	
Number of LUT Flip Flop pairs used	837			
Number with an unused Flip Flop	654	837	78%	
Number with an unused LUT	0	837	0%	
Number of fully used LUT-FF pairs	183	837	21%	
Number of slice register sites lost to control set restrictions	0	54,576	0%	
Number of bonded IOBs	8	296	2%	
Number of LOCed IOBs	8	8	100%	
Number of RAMB16BWERs	0	116	0%	
Number of RAMB8BWERs	0	232	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	0	16	0%	
Number of DCM/DCM_CLKGENs	0	8	0%	
Number of ILOGIC2/ISERDES2s	0	376	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	376	0%	
Number of OLOGIC2/OSERDES2s	0	376	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	256	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	48	58	82%	

Fig. 16 Second report of final results for implementing the proposed neural network in the FPGA.

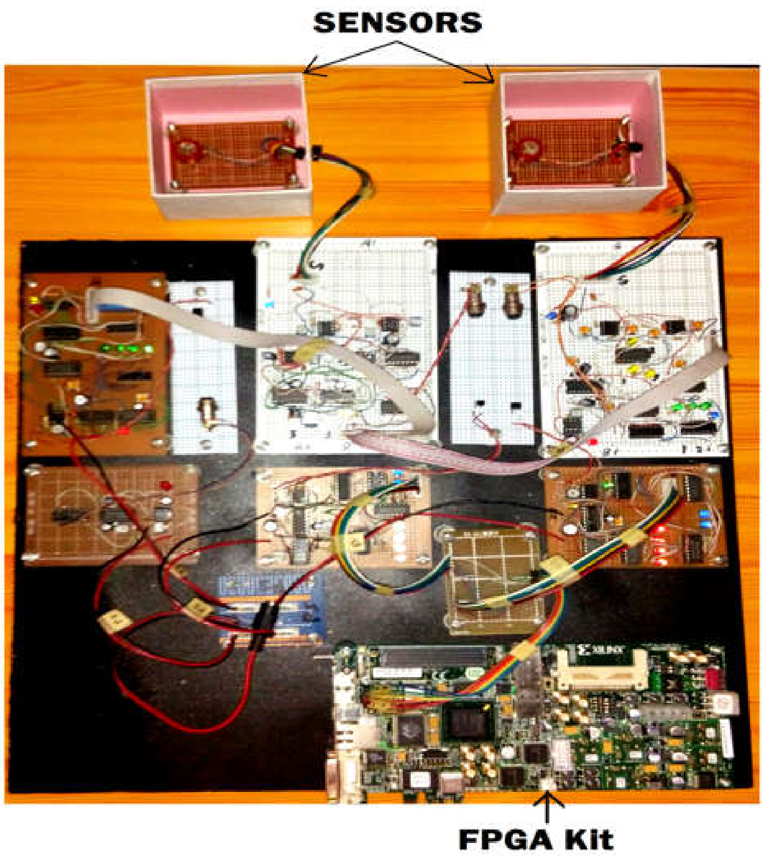


Fig. 17 Implementation of the related CO wireless sensor network which is connected to the FPGA.

The proposed system can be implemented using fast logic combinational circuit (Hardware) with the following characteristics: 1) the bit rate can reach to more than 550 Mbits/sec, which is fast than that of this work, 2) the consumed power nearly equals to that of this work, 3) the overall system can not be upgraded, 4) small size of building. Also, the proposed system can be built using low power microprocessor system (software with hardware) with the following characteristics: 1) The bit rate of this system is so slower than that of this work, which depends on low clock frequency of the microprocessor and low speed of the used RAM and ROM of the system, 2) the consumed power is so lower than that of this work, 3) the software of this system can be upgraded, 4) small size of building. Finally, the proposed system can be built using high speed microprocessor system (software with hardware) with the following characteristics: 1) The bit rate of this system is higher than that of this work, which depends on the high clock frequency of the microprocessor and high speed of the used RAM and ROM of the system, 2) the consumed power is higher than that of this work, 3) the software of this system can be upgraded, 4) large size of building. Table 2 illustrates the characteristics of the various systems that can be used by the proposed system in this work.

Table 2. Comparison among various systems which can used by the proposed work.

The proposed system	Bits Rate (speed)	Power Consumption	Capability to Upgrade	Size
In this work	200 Mbits/sec	Low	Yes	Small
Using fast logic combinational circuit	>550 Mbits/sec	Low	No	Small
Using low power microprocessor System	< 200 Mbits/sec	so lower than in this work	Yes	Small
Using high speed microprocessor system	> 200 Mbits/sec	Higher than in this work	Yes	Large

5. CONCLUSIONS

The sensor nodes of the Wireless Sensor Network WSN can be increased for the proposed work, in this case the input lines of the proposed simulation system must be increased, which leads to increase the neurons of the input layer of the proposed PBNN, so the size of the proposed software system will be enlarged. For this reason, A suitable FPGA must be chosen for implementing the upgraded simulation system, where, the maximum software size of the FPGA must be considered.

An essential condition must be considered for successful implementing of the proposed simulation system in the FPGA, which is the using of linear activation functions in hidden and output layers of the proposed neural network, whereas, using of non-linear activation functions leads to failure of implementing the proposed simulation system in the FPGA.

For increasing the accuracy of the output of the proposed system, one can increase the output levels of the proposed system, which means increasing the neurons of the output layer of the proposed BPNN, in this case the size of the proposed simulation system will be enlarged, again, the program size capacity of the required FPGA must be considered.

One can obtain optimal result by reaching the Mean-Square-Error MSE to 'zero' value using:1) 'Traingda' training function as a learning method for the proposed BPNN, 2) 'Satlins' and 'Satlin' linear activation functions for the hidden and output layers of the proposed BPNN. In fact, Increasing the hidden layers or decreasing the neurons of single hidden layer leads to undesirable results.

REFERENCES

[1] Ado A. ABBA, Abdelhak G., Nabila L., and Blaise O. Yenke, **Concepts and Evolution of Research in The Field of wireless Sensor Networks**, International Journal of Computer Networks and Communications (IJCNC), Vol.7, No.1, January, 2015.

[2] Utz R., and Cormac J. S., **Wireless Sensor Networks**, Springer-Verlog Berlin Heidelberg, 2009.

[3] Hanan A. R. Akkar, Aied K. Al-Samarrie, and Azzad B. Saeed, **Design and Implementation of an Interface Unit Communicated by a Laser system Within Wireless Sensor Network**, Engineering and Technology Journal, Vol.34,Part (A), No.13 ,2016.

[4] Shashi S, **A Gentle Introduction to Backpropagation**, Numeric Insight Inc., 22th, July, 2014.

[5] Yang L., Weizhe J., and Lixiong X. **Parallelizing Backpropagation Neural Network Using MapReduce and Cascading Model**, Yang Liu et al., 2016.

[6] Reyadh S. N., Namh A. A., and Zainab N. Al-Sultani, **An Enhanced Resilient Backpropagation Artificial Neural Network for Intrusion Detection System**, International Journal of Computer Science and Network Security (IJCSNS), VOL.12 No.3, March, 2012.

[7] Dian P., Diaz D. Santika, and Bens P., **An Application of Backpropagation Artificial Neural Network Method for Measuring The Severity of Osteoarthritis**, International Journal of Engineering and Technology IJET-IJENS Vol. 11 No. 03, June, 2011.

[8] Hanan A. R. Akkar, Aied K. Al-Samarrie, and Azzad B. Saeed, **Simulation Design of a Backpropagation Neural System of Sensor Network Trained by Particle Swarm Optimization**, International Journal of Scientific and Engineering Research IJSER, Vol.7, Issue 4, ISSN:2229-5518, April, 2016.

[9] F. Acar Saraci, **Artificial Intelligence and Neural Networks**, Springer-Verlag Berlin Heidelberg, 2006.

[10] Daniel S., Ian C., Daming S., and Wing W., **Sensitivity Analysis for Neural Networks,** Springer-Verlog Heidelberg, 2010.

[11] David L., **A Basic Introduction to Feed-forward Back-propagation Neural Networks**, David Leverington, 2009.

[12] K. Henkel, and D. Schmeiber**, Back-propagation- Based Neural with a Two Sensors System for Monitoring Carbon Dioxide and Relative Humidity**, Springer - Verlag, ISSN:1618-1650, 2002.

[13] Muhammed K., B. Muhammed K., and T. Muhammed J., **FPGA Based Neural Wireless Sensor Networks**, The 13th International Arab Conference on Information Technology, 2012.

[14] Guo W., and Juan W., **Design and Implementation of Wireless Sensor Network Nodes Based on BP Neural Network**, Journal of chemical and Pharmaceutical Research, ISSN:0975-7384, 2014.

[15] Azzad B. Saeed, **Simulation Design of an Intelligent System for Automotive Transmission Gearbox Based on FPG**A, EMITTER- International Journal of Engineering Technology, Vol. 6, No. 2, ISSN: 443-1168, December, 2018.

[16] Mark H.B., Martin T. H., and Haward B. D., **Neural Network Toolbox: User Guide**, The Math Works Inc., 2014.