International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-2, December 2019

Development of Virtual Flow Sensor using Artificial Neural Networks

Gangiregula Subba Rao, Venkata Lakshmi Narayana K., Naveen Kumar V., Pranav Sanghavi

Abstract*: In this paper, a virtual flow sensor using artificial neural networks (ANN) is proposed to improve the efficiency of an industrial flow control loops. In conventional flow-control loop, flow meters used for sensing flow rate in the feedback path cause pressure drop in the flow. This may increase the energy usage for propelling the fluid. The functional relation between the flow rate and the physical properties of the flow through the final control element such as control valve is known and the said properties namely pressure drop, temperature, and valve position are yielded from an experimental set-up. These properties are used as training data for ANN models to yield the fluid flow rate through the control valve. Here, the ANN acts as a virtual flow sensor. The feasibility of the proposed methodology is validated by using real measurement of flow and used them to model virtual flow sensor using the multi-layer perceptron artificial neural networks (MLP-ANN) with back propagation (BP) algorithm. Moreover, its practical proof of concept is demonstrated by implementing the trained MLP-ANN on a Spartan-3E-starter Field Programmable Gate Array (FPGA) unit through a hardware co-simulation.

Keywords: Artificial neural networks, back propagation algorithm, control valve, flow-rate, multi-layer perceptron, virtual sensor.

I. INTRODUCTION

In process-control industries, measurement and control of fluid flow-rate is an important task. In a typical industrial control loop, the output quantity of a flow sensor corresponding to input flow measured is compared with the set-point. According to the error signal of the comparator, the controller commands the actuator mainly the flow-control valve to force the actual flow close to the set point. Generally, low cost flow sensors such as venture- meters and turbine flow meters are employed to measure the volumetric fluid flow in unit time [1]. But these flow sensors cause pressure drop in the flow of fluids and results in the consumption of more energy for pumping the fluids.

Revised Manuscript Received on December 30, 2019.
***** Correspondence Author
Gangiregula Subbarao, Department of Electronics & Communications Engineering, Sri SatyaSai University of Technology and Medical Sciences, Sehore, India. Email: subbu.electronics17@gmail.com
Venkata Lakshmi Narayana K., School of Electrical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India. Email: vlnarayana.k@vit.ac.in
Naveen Kumar V.*, School of Electronics Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India. Email: vegenaveen@vit.ac.in
Pranav Sanghavi, Department of Computer Science and Electrical Engineering, West Virginia University, Morgan Town, West Virginia, USA. Email: pranav.sanghavi@mail.wvu.edu

This problem can be overcome by using non-contact flow sensors such as electromagnetic flow sensors which doesn't obstruct the flow of fluids. The non-contact flow sensors have been designed and vastly used in instrumentation and control industries as there is no necessity for installation in the pipeline. Also introduction to the differential pressure across pipelines is not necessitated. The high cost of non-contact flow sensors restricts its use than that of its conventional sensors.

In this paper, an ANN based virtual flow sensor is proposed to conduct the flow measurement of fluids for flow-control valves. The main intention is to eliminate the pressure drop and the associated power loss that occur in basic flow sensors. The virtual flow sensor is designed as an alternative to high cost non-contact flow sensors. The principle approach is to determine the relation between the flow rate and physical properties of the control valve for flow measurements. In this approach, the physical characteristics and properties of the control valve are embedded in MLP-ANN model to determine the flow rate of fluids instead of real measurement using a conventional flow sensor.

Neural networks have been extensively used to add intelligence to sensors. Linearization [2], self-calibration [3] or extension of the range of the sensor is also performed using ANNs in various studies [4]. The universal approximation properties of feed forward neural networks allow us to create virtual sensors whose output through the neural network model would supply the output as if it were the physical sensors without the parametrical losses and investment of the said sensor [3]. A neural network model is preferred over statistical methods on accounts of outperforming including capabilities to learn from exemplars, plasticity of architecture, parallel computation, fault tolerant, robustness to noise and ease of implementation on hardware environments such as the FPGA.

As with most hardware implementations, digital or analog paradigms are considered to realize ANNs on a hardware device. The digital implementation is more prevalent in practical adoption due to high accuracy, high repeatability, better noise immunity, testability, advanced reliability and flexibility with all types of processing systems. Conversely, analog implementation is more problematic in design and can only be viable for specific domains and applications involving mass production. The ANN hardware implementations can be performed by application specific integrated chips (ASIC), digital signal processing modules (DSP) and FPGA units [5]. The ASIC implementation of ANN models does not offer re-configurability by the user. The DSP modules do not preserve the parallel topology of the neurons in a layer as DSP implementation is sequential.

FPGA units preserve the parallel topology and architecture of the neurons in any ANN model and hence are considered to be the most suitable hardware implementation components.

Hence it is computationally viable and can be reconfigured by the user. FPGA implementations of feed forward networks have been implemented successfully in various studies [6]. A thorough activation function approximation has been made and the multilayer perceptron is implanted on the Spartan-3E FPGA starter unit to obtain the proposed objective. The paper is organised as follows. The organization of the paper is as follows. The mathematics of the control valve is described in section II. The steps involved in the development of MLP-ANN using BP algorithm are elaborated in section III. The FPGA implementation of ANN-MLP using BP algorithm and results are presented in section IV and the conclusion is given in section V.

II. THEORY OF THE CONTROL VALVE

The process control and instrumentation systems are humungous domains. Mainly the process control applications range from simple domestic cooker control to a whole production unit or process. Various physical quantities such as temperature, pressure, humidity, flow, density, level need to be continuously controlled and monitored [7]. One of the most important aspects of such control is typically the control of flow of fluid and liquids using control valves. In this work, an equal percentage valve is employed. The flow through the valve is given by

$$Q = C_v f(\theta)\sqrt{\frac{\Delta P}{G}} \tag{1}$$

where Q represents the flow rate, C_v indicates the valve coefficient, G represents the specific gravity of the fluid and the pressure drop across the flow-control valve is indicated by ΔP. $f(\theta)$ is defined by

Linear valve: $f(\theta) = \theta$ (2)

Quick opening valve: $f(\theta) = \sqrt{\theta}$ (3)

Equal percentage valve: $f(\theta) = R^{(\theta-1)}$ (4)

where, $f(\theta) \in (0,1)$ and θ is the lift of the valve. It is the relative position of the valve globe with reference to the valve seat. R is the valve design parameter $R \in (20,50)$ which depends on the geometry of the valve employed. According to (1), the flow rate through the control valve is a function of the differential pressure and lift. It can be rephrased mathematically without loss of generalization that,

$$Q = f(\Delta P, \%) \tag{5}$$

Where, ΔP is the differential pressure and % is the percentage opening of the valve which also indicates the stem position.

III. MULTILAYER PERCEPTRON NEURAL NETWORK

In the MLP-ANN, the input signals propagate in a forward direction, from input layer to hidden layer and to output layer respectively. The MLP-ANN has been used in sensor linearization and compensation applications with the back-propagation algorithm [9].

For the network, shown in Fig. 1, the feed-forward output is given as,

$$y = g\left(\sum\left(f\left(\sum I_i.v_{ij} + b_j\right).w_k\right) + b_{k'}\right) \tag{6}$$

where $b_{k'}$, w_k and b_j, v_{ij} are the bias' and weights to the output and hidden layer respectively and $g(.)$ and $f(.)$ are the activations functions in the output and hidden neurons respectively. The weights and bias' are updated as per the backpropagation algorithm as illustrated by the flow chart in Fig. 2.

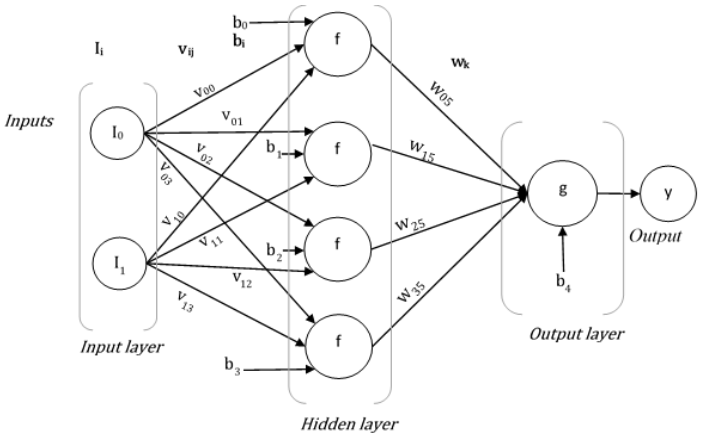

Fig. 1 Multilayer perceptron feed-forward neural network.

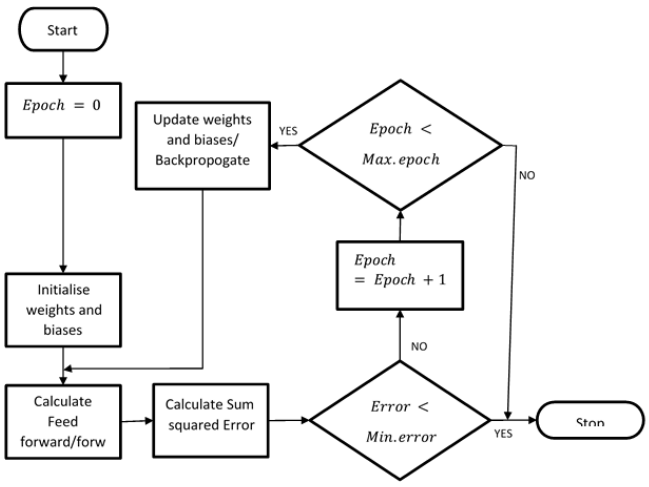

Fig. 2 Flow chart of back propagation algorithm

The back propagation algorithm as discussed consists of two distinct steps:

Forward pass: The network shown in Fig. 1 gives the output as according to (6).

Back propagation: In this step there is the apparent computation in the backwards direction.

The target data is given as *y'*, hence the error is

$$e = y' - y \qquad (7)$$

For the output layer, the error is

$$\delta_0 = e.[y(1-y)] \qquad (8)$$

For the hidden layer, the error is

$$\delta_h = \left(\sum \delta_0 . w\right) f' \qquad (9)$$

where f' are the actual outputs in the hidden layer. This is followed by the updation of weights:

For the output layer:

$$w_k(new) = w_k(old) + \alpha\delta_0 f \qquad (10)$$

$$b_{k'}(new) = b_{k'}(old) + \alpha\delta_0 \qquad (11)$$

For the hidden layer:

$$v_{ij}(new) = v_{ij}(old) + \alpha\delta_0 f \qquad (12)$$

$$b_j(new) = b_j(old) + \alpha\delta_0 \qquad (13)$$

where α is the learning rate and *f* is hidden layer output.

Using these tools the neural network is constructed to emulate a flow sensor. As illustrated in equation (5), the proposed artificial neural network requires 2 inputs namely the differential pressure and stem position of valve and one output, the flow. A dataset of these parameters are carefully obtained experimentally.

Table- I: ANN parameters

Neural network	Multilayer Perceptron	
Number of hidden neurons	4	
Activation function	Hidden neurons - log-sigmoid, Output neuron - linear	
Training algorithm	Backpropagation	
Epochs	1000	
Input weights to hidden neurons	V_{00}=1.0522 V_{01}=3.0286 V_{02}=-0.9959 V_{03}=2.0069	V_{10}=3.6616 V_{11}=2.0187 V_{12}=-0.0884 V_{13}=-4.2753
Input biases to hidden neurons	b_0 = -0.3798, b_1 = 0.3076, b_2 = 0.3099, b_3 = -0.3674	
Weights to output neuron	w_{05} = 5.1865, w_{15} = -3.0431, w_{25} = -3.9924, w_{35} = 2.3473	
Bias to output neuron	b_4 = -0.1034	

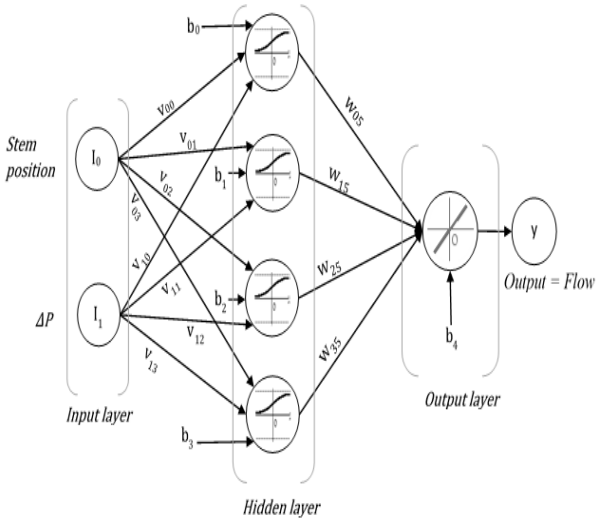

Fig. 3 MLP network to approximate flow rate

The back-propagation (BP) algorithm with a learning rate parameter of 0.3 is implemented for training the MLP. The stem position of the valve and the differential pressure are normalized and are considered as inputs to the MLP-ANN. Similarly, the flow through control valve is normalized and is considered as the target for the ANN. All the inputs and target are applied and the BP algorithm updates the weights and biases of MLP-ANN. The final updated weights are shown in Table I and the performance of the network in terms of error in each epoch over all epochs is plotted in Fig. 4. The optimum weights and biases of the MLP-ANN are stored for testing, validation and further processing. During the testing and validation process, the stored weights are applied into the MLP-ANN. The test inputs are fed to this MLP-ANN to validate the output. The output *i.e.*, the predicted flow is compared with the expected flow rate and is illustrated in Fig. 5.

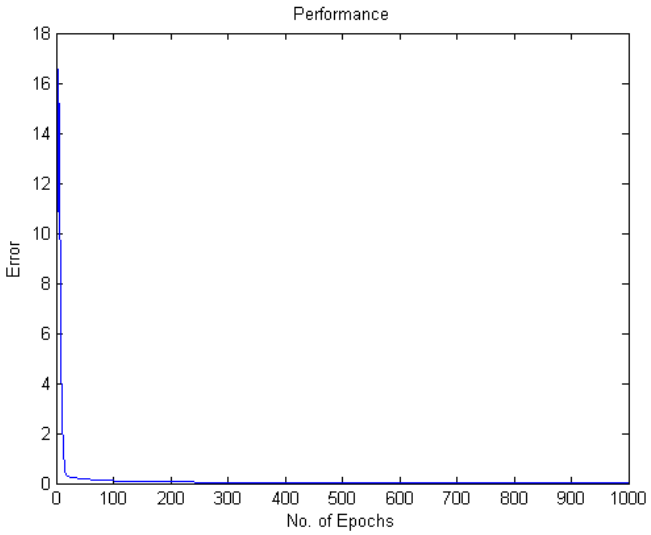

Fig. 4 Performance of MLP-ANN in terms of error over subsequent epochs for all epochs

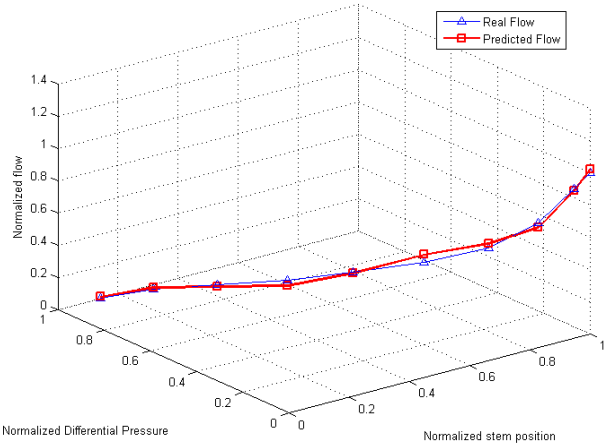

Fig. 5 Valve characteristics: Expected & predicted from MLP- BPNN direct modelling

For optimum training of the neural network, these values are normalized. The MLP network with 4 hidden neurons, with log-sigmoid function and linear activation functions respectively, as shown in Fig. 3 is coded using MATLAB.

IV. FPGA IMPLEMENTATION AND RESULTS

FPGAs are reprogrammable digital ICs that were developed in the mid 1980's. A personal computer can be used to design a digital circuit which is then compiled into a special programming file and it will realize the circuit once it is downloaded to the FPGA. FPGA, available from the manufactures, currently consists of thousands of reconfigurable logic elements. Each logic element typically contains some number of D-flip-flop, look-up tables, NAND gates, and NOR gates. The wiring that connects these components, as well as the wiring that connects neighboring logic elements, is electrically reconfigurable. Therefore, components can be wired together to create one design. Then it can be erased and wired together differently to create a new one.

In this study the FPGA design is accomplished with the help of the Xilinx System Generator toolbox [8]. It consists of various blocks of specific functions that are required and interlinked to compose algorithms. These blocks do not only serve to simulation, but can also generate VHDL or Verilog code. In addition, the library provided by Xilinx to Simulink, contain main blocks that are basic for the ANN except a few functions such as sigmoid function. We perform both a purely software simulation on Simulink as well as a hardware/software co-simulation through the JTAG emulator generated by the Xilinx System Generator toolbox.

A. *FPGA implementation of the sigmoid function*

The log-sigmoid function consists of an infinite exponential series and cannot be implemented directly [10]. The simplified version of log- sigmoid function is considered for computational implementation. There are several methods to approximate the said function: Look-up tables, piecewise linear approximation among others. [5] Other study employs the Taylor series approximation [6] truncated to the second order terms. The log-sigmoid activation function is:

$$f(x)=\frac{1}{\left(1+e^{-x}\right)} \tag{14}$$

The Taylor series expansion of any function $F(x)$ in its defined domain can be given as:

$$F(x)= k_0+k_1x+k_2x^2+k_3x^3+k_4x^4+\mathrm{K} \tag{15}$$

The log-sigmoid function is asymptotic between the lines $y=0$ and $y=1$. The asymptotic distance between the function and the asymptotes becomes negligible in the range $(-\infty,-4)\cup(4,\ \infty)$. Thus the function $f(x)$ is linear in the range $(-\infty,-4)\cup(4,\ \infty)$ with the value equal to that of the asymptotes. The Taylor series can be truncated to the second degree term to reduce complexity by minimizing the required number of multipliers to implement the function in FPGA. So now we have the Taylor series as:

$$F(x)= a_0+ a_1x+ a_2x^2 \tag{16}$$

The constants a_0, a_1, a_2 can be easily calculated and the log-sigmoidal function is expressed as:

$$f(x)=\begin{cases} 0 \ \ \forall x\in(-\infty,\ -4] \\ 0.5+0.2543x+0.0332x^2 \ \ \forall x\in(-4,\ 4) \\ 1 \ \forall x\in[4,\ \infty) \end{cases} \tag{17}$$

Referring to the equation (17), the log-sigmoid block is constructed using the Xilinx sysgen library in Simulink as shown in Fig. 6.

B. *FPGA implementation of the MLP network*

FPGA implementation requires a series of interconnected multipliers and adders to construct so as to resemble the MLP network. This has been illustrated in Fig. 7. The activation function, log-sigmoid, has to be constructed as derived and exhibited earlier.

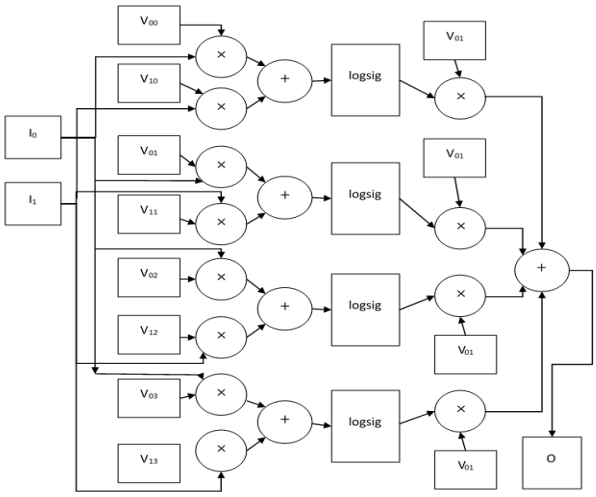

Fig. 6 Block diagram of the MLP network made up entirely of adders and multipliers and the activation function

The system general library block is placed as seen in Fig. 8. and the necessary sampling rates are applied for the system. Care has to be taken while assigning sampling rates as it should satisfy the Nyquist sampling theorem and the FPGA board maximum frequency. The Gateway in blocks samples, quantizes and discretizes the input data and send it to the other blocks, which now act as components of an FPGA environment subject to necessary constrains. Gateway out block again converts data to be readable by MATLAB; in this system manually input the test data one after the other recording the respective output. This simulation gives the tentative outputs to expect after dumping this model on the FPGA board.

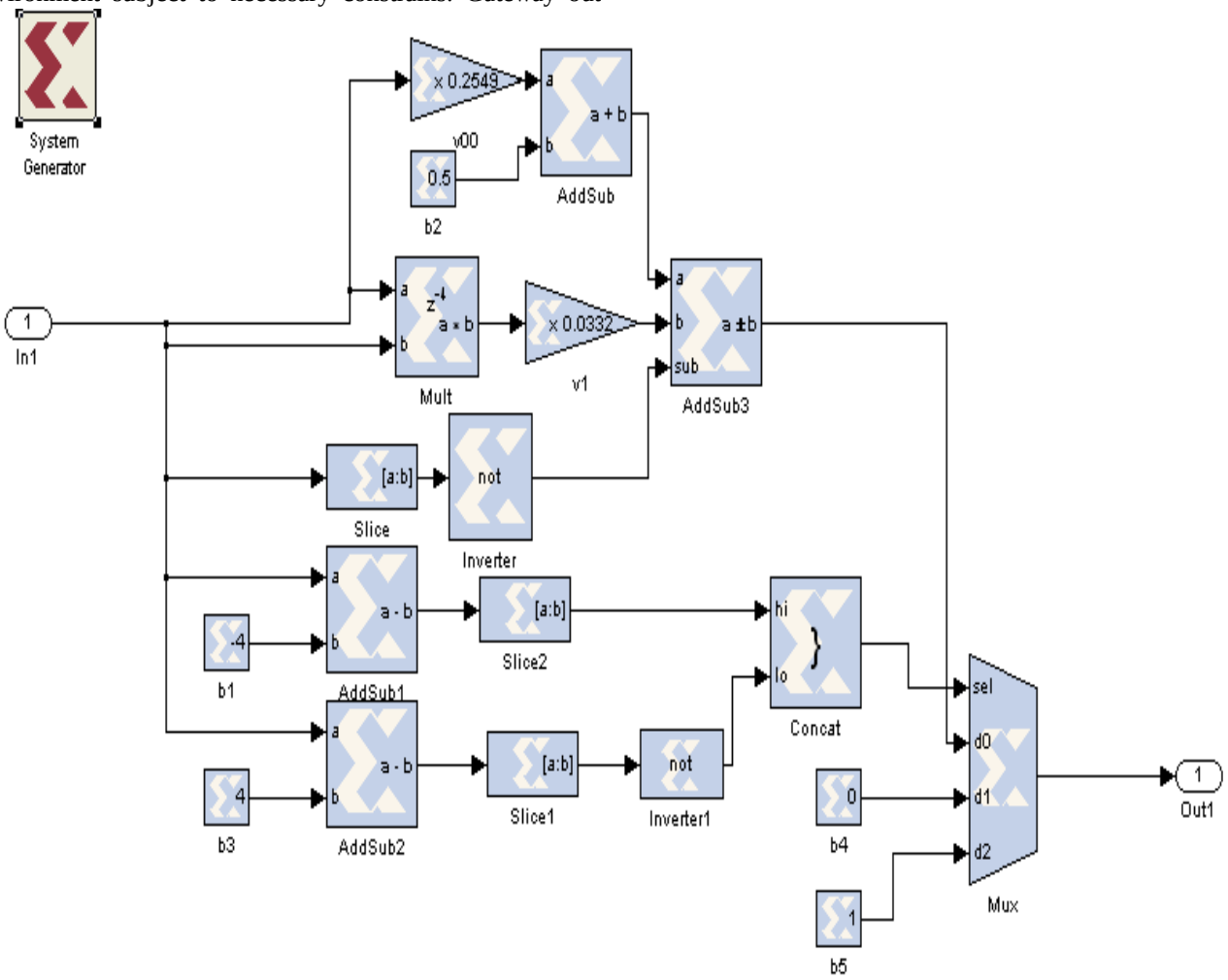

Fig. 7 Simulink model - Taylor series approximation of the log-sigmoidal function using Xilinx sysgen blocks

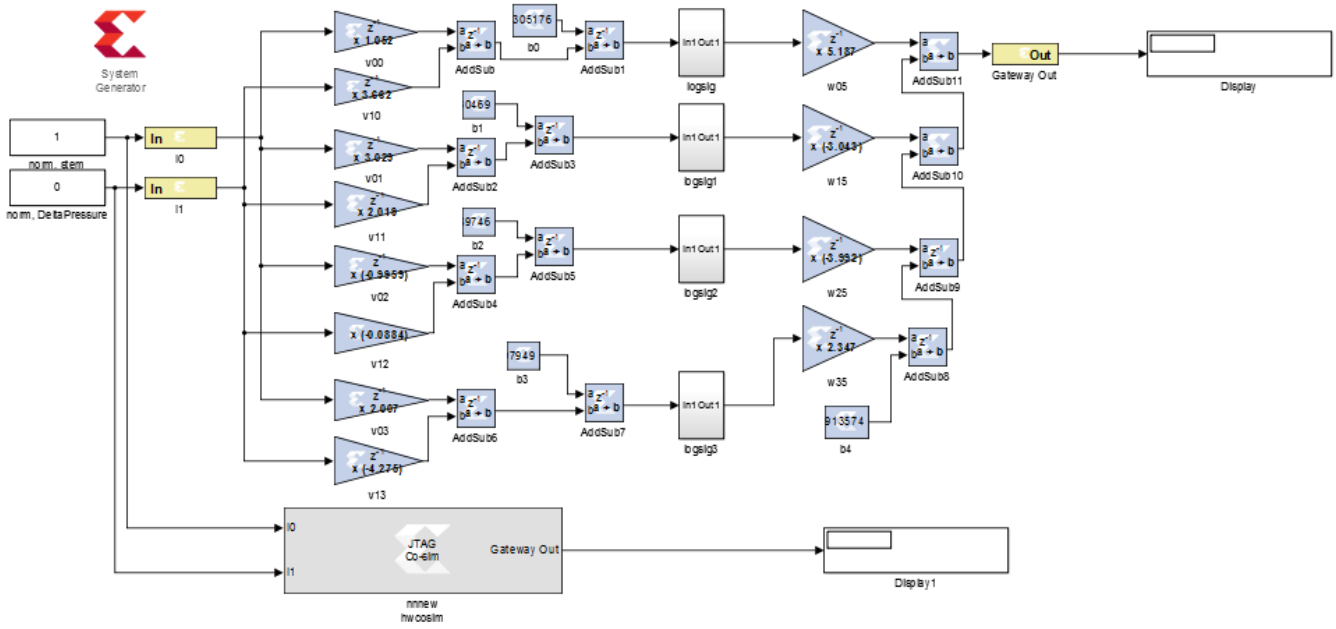

Fig. 8 Xilinx Sysgen model of the trained MLP network along with the JTAG – emulator block for hardware co-simulation

Now this model is dumped onto an FPGA board. Sysgen allows converting this block into verilog and/or VHDL code ready for implementation in FPGA. Next step is to perform a hardware co-simulation of the model via compiling a JTAG – emulator block. The sysgen blockset, after compiling the VHDL code of the blockset in Fig. 8 occupies just 48.9% of space in the FPGA, constructs the necessary bitstream file for communication between the computer and the FPGA board. The Spartan 3E starter kit is configured as the target co-simulation device and the block is created. The output is then recorded.

V. CONCLUSION

The developed work has two distinct sections namely, the neural network training and simulation and the FPGA implementation of the MLP-ANN with BP algorithm. The BP algorithm for training of MLP-ANN is coded in MATLAB and the updated weights are used to test the network. The output from the training data after 1000 epochs has the sum of squared error of 0.0092 and the output from the test data offered a mean squared error of 4.306×10^{-4}. The FPGA implementation is accomplished through the Xilinx system generator blockset. The Simulink model of the neural network using Xilinx System Generator is set and the simulation on Simulink is set up; the Simulink output and the FPGA hardware board co-simulation output has the mean square error with respect to the expected flow as 0.0057 and 48.9% board space utilization. The study can further include the live interfacing of the sensors from the control valve with the FPGA board. Moreover, the back-propagation algorithm itself can be implemented on the FPGA with the device to be used to its peak flexibility. This study provides yet another experimental validation of the universal approximation theorem pertaining to neural networks and can thus be used for several other systems, by substituting expensive sensors.

REFERENCES

1. P. Pallás-Areny and J. G. Webster, Sensors and Signal Conditioning, 2nd ed. New York, NY, USA: Wiley, 2013.
2. Vaegae Naveen Kumar and K. V. Lakshmi Narayana, “Development of an intelligent pressure sensor with temperature compensation” Journal of Engineering Science and Technology,. vol. 12, no.7, pp. 1723-1739, 2017.
3. J. C. Patra, A. C. Kot, and G. Panda, “An intelligent pressure sensor using neural networks,” IEEE Trans. Instrum. Meas., vol. 49, no. 4, pp. 829–834, Aug. 2000.
4. V. Naveen Kumar and K. V. Lakshmi Narayana, “Development of thermistor signal conditioning circuit using artificial neural networks,” IET Sci., Meas. Technol., vol. 9, no. 8, pp. 955–961, Nov. 2015.
5. H. S. A. D. and M. A. , “Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization,” IEEE Transactions on Neural Networks, vol. 18, no. 3, pp. 880-888, 2007.
6. S. A. Al-Kazzaz and R. A. Khalil, “FPGA Implementation of Artificial Neurons: Comparison Study,” 3rd International Conference on Information and Communication Tchnologies: from Theory to Applications, 7-11 April, 2008.
7. D. E. Seborg, T. F. Edgar and D. A. Mellichamp, Process Dynamics and Control, 3rd ed. John Wiley & Sons, 2011.
8. Xilinx Inc., “Xilinx, "Spartan™-3E Platform FPGAs Complete Data Sheet", Datasheet Sheet DS312,” 2013.
9. B. Bavarian, “Introduction to Neural Netwroks and Control,” Special Section on Neural Networks for Systems and Control, April 1988.
10. D. E. Khodja, A. Kheldoun and L. Refoufi, “Sigmoid Function Approximation for ANN Implementation in FPGA devices,” Recent Researches in Circuits, Systems, Electronics, Control & Signal Processing. pp. 213–235, 2010.

AUTHORS PROFILE

Gangiregula Subbarao received the M.Tech. degree in Communication Systems from S.R.M University, Chennai, India in 2014. He is currently pursuing Ph.D. degree at department of electronics & communications engineering, Sri Satya Sai University of Technology and Medical Sciences, Sehore, India from 2016. His interests are in the area of sensors and signal conditioning, Wireless communications.

K. Venkata Lakshmi Narayana received the B.Tech. degree in instrumentation engineering from the Nagarjuna University, Guntur, India in 2001, and the M.Tech. degree in electrical engineering from the Motilal Nehru National Institute of Technology (MNNIT), Allahabad, India in 2006, and the Ph.D. degree in instrumentation engineering from the Andhra University, Visakhapatnam, in 2013. He has authored more than 30 research papers in refereed journals and over 15 papers in conference proceedings. He is currently a Senior Associate Professor at the School of Electrical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India. His interests are in the area of sensors and signal conditioning, measurements, wireless sensor networks, optimization, process instrumentation and virtual instrumentation.

V. Naveen Kumar received the B.Tech. degree in instrumentation engineering from the Nagarjuna University, Guntur, India in 2001, and the M.Tech. degree in electronics engineering from the Jawaharlal Nehru Technological University, Kakinada, India in 2010 , and the Ph.D. degree in electrical and electronics engineering from the Vellore Institute of Technology, Vellore in 2016. He is currently an Associate Professor at the School of Electronics Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India. He has authored over 15 research papers in refereed journals and over 9 papers in conference proceedings. His interests are in the area of sensors and signal conditioning, signal processing, neural networks and fuzzy logic systems.

Pranav Sanghavi received the B.Tech. degree in electronics and instrumentation engineering from the Vellore Institute of Technology, Vellore, India in 2014. He is currently pursuing Masters degree with graduate research assistantship at West Virginia University, Morgan Town, West Virginia from 2016. His interests are in the area of sensors and signal conditioning, fast radio bursts, RET-DSPIRA, astrophysics and soft computing.