IJCSN International Journal of Computer Science and Network, May 2014
ISSN (Online) : 2277-5420
www.IJCSN.org

Implementation of Neuro-Fuzzy System Using VHDL

[1] K. L. Kar, [2] Shamita Chakraborty, [3] B.B. Mangaraj

[1] Asso. Prof., Department of Electronics and Telecommunication Engineering
MMCT, Raipur, C.G. India.

[2] Director, MMCT, Raipur, C.G. India

[3] Asso. Prof., Department of Electronics Engineering
VSSUT, Burla, Odisha India

Abstract - In theory, Artificial Neural Networks (ANN) and fuzzy systems can interchange their position in hybrid system, yet in practice each type has its own advantages and disadvantages. In case of ANNs, the knowledge is automatically acquired by the back-propagation algorithm, but the learning process is relatively slow and analysis of the trained network is difficult. Further, it is not possible to extract structural knowledge (rules) from the trained ANN and also not possible to integrate special information about the problem into the ANN in order to simplify the learning procedure. Fuzzy systems are more favorable in a sense that their behavior can be explained based on fuzzy rules and thus their performance can be adjusted by modifying the rules. But, since in general, knowledge acquisition is difficult and also the universe of discourse of each input variable needs to be divided into several intervals, applications of fuzzy systems are restricted to the fields where expert knowledge is available and the number of input variables is not large. Fusion of ANN and Fuzzy Inference Systems (FIS) have attracted the growing interest of researchers in various scientific and engineering areas due to the growing need of adaptive intelligent systems to solve the real world problems[1]. ANN learns from scratch by adjusting the interconnections between layers. This paper starts with a discussion of the features of each model and generalizes the advantages and deficiencies of each model and finally implements using VHDL[2].

Keywords - *Neuro-Fuzzy, VHDL.*

1. Introduction

Neuro Fuzzy (NF) computing is a popular framework for solving complex problems. While neural networks have strong learning capabilities at the numerical level, it is difficult for the users to understand them at the logic level. Fuzzy logic, on the other hand, has a good capability of interpretability and can also integrate expert's knowledge. The hybridization of both the paradigms yields the capabilities of learning, good interpretation and incorporating prior knowledge. The combination can be in different forms. The simplest form may be the concurrent neuro-fuzzy model, where a fuzzy system and a neural network work separately. The output of one system can be fed as the input of the other system. The cooperative neuro-fuzzy model corresponds to the case that one system is used to adapt the parameters of the other system [3], [4]. The hybrid neural-fuzzy model is the true synergy that captures the merits of both the systems. It takes the form of either a fuzzy neural network or a neuro-fuzzy system. A hybrid neural-fuzzy system does not use multiplication, addition, or the sigmoidal function, but uses fuzzy logic operations such as t-norm and t-conorm.

A fuzzy neural network [5] is a neural network equipped with the capability of handling fuzzy information, where the input signals, activation functions, weights, and/or the operators are based on the fuzzy set theory. Thus, symbolic structure is incorporated. The network can be represented in an equivalent rule-based format, where the premise is the concatenation of fuzzy AND and OR logic, and the consequence is the network output. Two types of fuzzy neurons, namely AND neuron and OR neuron, are defined. The NOT logic is integrated into the weights. Weights always have values in the interval [0,1], and negative weight is achieved by using the NOT operator. The weights of the fuzzy neural network can be interpreted as calibration factors of the conditions and rules. A neuro-fuzzy system is a fuzzy system, whose parameters are learned by a learning algorithm. It has a neural network architecture constructed from fuzzy reasoning, and can always be interpreted as a system of fuzzy rules. Learning is used to adaptively adjust the rules in the rule base, and to produce or optimize the MFs of a fuzzy system. Structured knowledge is codified as fuzzy rules. Expert knowledge can increase learning speed and estimation accuracy. Both fuzzy neural networks and neuro-fuzzy systems can be treated as neural networks, where the units employ the t-norm or t-conorm operator instead of an activation function. The hidden layers represent fuzzy rules. The line between the two hybrid models is blurred,

and we call both types of synergisms as neuro-fuzzy systems.

Neuro-fuzzy systems can be obtained by representing some of the parameters of a neural network, such as the inputs, weights, outputs, and shift terms as continuous fuzzy numbers. When only the input is fuzzy, it is a Type I neuro-fuzzy system. When everything except the input is fuzzy, we get a Type II model. A type III model is defined as one where the inputs, weights, and shift terms are all fuzzy. The functions realizing the inference process, such as t-norm and t-conorm, are usually non-differentiable. To utilize gradient-based algorithms, one has to select differential functions for the inference functions. For non-differentiable inference functions, training can be performed by using EAs. The shape of the MFs, the number of fuzzy partitions, and rule base can all be evolved by using EAs. The neuro-fuzzy method is superior to the neural network method in terms of the convergence speed and compactness of the structure. Fundamentals in neuro-fuzzy synergism for modeling and control have been reviewed in [6].

2. Integration of Fuzzy Logic and Neural Networks

Hybrid systems combining fuzzy logic, neural networks, genetic algorithms, and expert systems are proving their effectiveness in a wide variety of real-world problems. Every intelligent technique has particular computational properties (e.g. ability to learn, explanation of decisions) that make them suited for particular problems and not for others. For example, while NNs are good at recognizing patterns, they are not good at explaining how they reach their decisions. Fuzzy logic systems, which can reason with imprecise information, are good at explaining their decisions but they cannot automatically acquire the rules they use to make those decisions. These limitations have been a central driving force behind the creation of intelligent hybrid systems where two or more techniques are combined in a manner that overcomes the limitations of individual techniques. Hybrid systems are also important when considering the varied nature of application domains. Many complex domains have many different components of problems, each of which may require different types of processing. If there is a complex application which has two distinct sub-problems, say a signal processing task and a serial reasoning task, then a NN and an expert system respectively can be used for solving these separate tasks. The use of intelligent hybrid systems is growing rapidly with successful applications in many areas including process control, engineering design, financial trading, credit evaluation, medical diagnosis, and cognitive simulation.
While fuzzy logic provides an inference mechanism under cognitive uncertainty, computational NNs offer exciting advantages, such as learning, adaptation, fault-tolerance, parallelism and generalization. To enable a system to deal with cognitive uncertainties in a manner more like humans, one may incorporate the concept of fuzzy logic into the neural networks.

The computational process envisioned for fuzzy neural systems is as follows. It starts with the development of a "fuzzy neuron" based on the understanding of biological neuronal morphologies, followed by learning mechanisms. This leads to the following three steps in a fuzzy neural computational process :first development of fuzzy neural models motivated by biological neurons, second models of synaptic connections which incorporate fuzziness into neural network, third development of learning algorithms (that is the method of adjusting the synaptic weights)[7].

3. Types of Fuzzy Neural Systems

In response to linguistic statements, the fuzzy interface block provides an input vector to a multi-layer neural network. The NN can be adapted (trained) to yield desired command outputs or decisions.

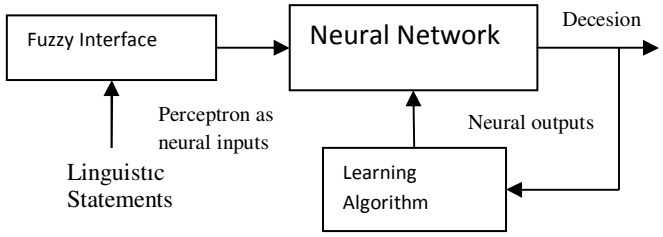

Figure 1: The first model of fuzzy neural system.

A multi-layered NN drives the fuzzy inference mechanism.

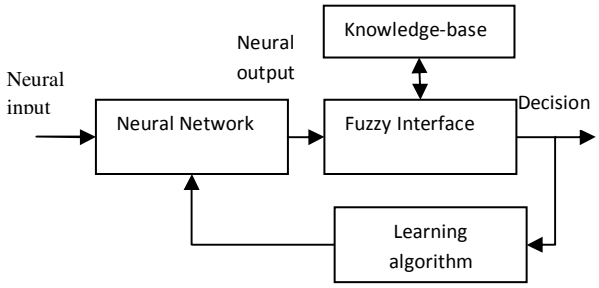

Figure 2: The second model of fuzzy neural system.

NNs are used to tune membership functions of fuzzy systems that are employed as decision-making systems for controlling equipment. Although fuzzy logic can encode expert knowledge directly using rules with linguistic labels, it usually takes a lot of time to design and tune the membership functions which quantitatively define these linguistic labels. NN learning techniques can automate this process and substantially reduce development time and cost while improving performance.

In theory, NNs, and fuzzy systems are equivalent in a sense that they are convertible, yet in practice each has its own advantages and disadvantages. For neural networks, the knowledge is automatically acquired by the BP algorithm, but the learning process is relatively slow and analysis of the trained network is difficult. Neither it is possible to extract structural knowledge (rules) from the trained neural network, nor can we integrate special information about the problem into the NN in order to simplify the learning procedure. Fuzzy systems are more favorable in a sense that their behavior can be explained based on fuzzy rules and thus their performance can be adjusted by tuning the rules. But since, in general, knowledge acquisition is difficult and also the universe of discourse of each input variable needs to be divided into several intervals, applications of fuzzy systems are restricted to the fields where expert knowledge is available and the number of input variables is small. To overcome the problem of knowledge acquisition, NNs are extended to automatically extract fuzzy rules from numerical data.

Cooperative approaches use NNs to optimize certain parameters of an ordinary fuzzy system, or to preprocess data and extract fuzzy (control) rules from data. Based upon the computational process involved in a fuzzy-neuro system, one may broadly classify the fuzzy neural structure as feed-forward (static) and feedback (dynamic).

4. Hybrid Neural Net

A hybrid neural net is a neural net with crisp signals and weights and crisp transfer function. However, (a) we can combine x_i and w_i using a t-norm, t-conorm, or some other continuous operation. (b) we can aggregate p_1 and p_2 with a t-norm, t-conorm, or any other continuous function. (c) f can be any continuous function from input to output.

We emphasize here that all inputs, outputs and the weights of a hybrid neural net are real numbers taken from the unit interval [0; 1]. A processing element of a hybrid neural net is called fuzzy neuron. Following are fuzzy neurons.

4.1 AND Fuzzy Neuron

The signal x_i and w_i are combined by a triangular conorm S to produce

$p_i = S(w_i, x_i)$; i = 1,2; ……..……1

The input information p_i is aggregated by a triangular norm T to produce the output of the neuron.

$y = AND(p_1, p_2) = T(p_1, p_2) = T(S(w_1, x_1), S(w_2, x_2))$ …….2

So, if T = min and S = max then the AND neuron realizes the min-max composition

$y = \min \{w_1 \vee x_1, w_2 \vee x_2\}$……..……3

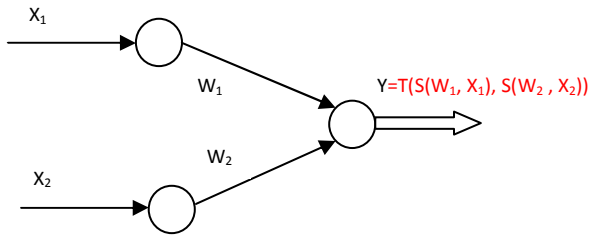

Figure 3: AND fuzzy neuron

4.2 OR (logic) Fuzzy Neuron

The signal x_i and w_i are combined by a triangular norm T to produce

$p_i = T(w_i, x_i)$; i = 1,2. ……..……4

The input information p_i is aggregated by a triangular conorm S to produce the output

$y = OR(p_1, p_2) = S(p_1, p_2) = S(T(w_1, x_1), T(w_2, x_2))$ ……5

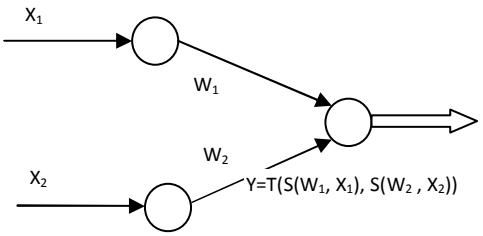

Figure 4: OR Fuzzy Neuron

So, if T = min and S = max then the AND neuron realizes the max-min composition

$y = \max\{ w_1 \wedge x_1;\ w_2 \wedge x_2\}$; ……...6

The AND and OR fuzzy neurons realize pure logic operations on the membership values. The role of the connections is to differentiate between particular levels of impact that the individual inputs might have on the result of aggregation. We note that (i) the higher the value w_i the stronger the impact of x_i on the output y of an OR neuron, (ii) the lower the value w_i the stronger the impact of x_i on the output y of an AND neuron. The range of the output value y for the AND neuron is computed by letting all x_i equal to zero or one. In virtue of the monotonicity property of triangular norms, we obtain

$y \in [T(w_1, w_2), 1]$ ……7

and for the OR neuron one derives the boundaries
$y \in [0; S(w_1, w_2)]$; ……8

4.3 Implication Of OR Fuzzy Neuron

The signal x_i and w_i are combined by a fuzzy implication operator I to produce

$p_i = I(w_i, x_i) = w_i \leftarrow x_i; \qquad i = 1,2;$ ……9

The input information pi is aggregated by a triangular conorm S to produce the output of the neuron.

$y = I(p_1, p_2) = S(p_1, p_2) = S(w_1 \leftarrow x_1; w_2 \leftarrow x_2);$ ……10

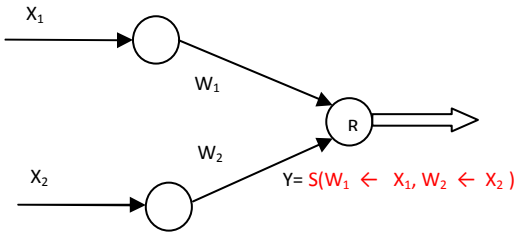

Figure 5: Implication-OR Fuzzy Neuron

Though hybrid neural nets cann't use directly the standard error BP algorithm for learning, they can be trained by steepest descent methods to learn the parameters of the membership functions representing the linguistic terms in the rules (supposing that the system output is a differentiable function of these parameters).The direct fuzzification of conventional NNs is to extend connection weights and/or inputs and/or fuzzy desired outputs (or targets) to fuzzy numbers. This extension is summarized in Table 1.

Table 1: Direct fuzzification of neural networks

Fuzzy neural net	Weights	Inputs	Targets
Type 1	crisp	fuzzy	crisp
Type 2	crisp	fuzzy	fuzzy
Type 3	fuzzy	fuzzy	fuzzy
Type 4	fuzzy	crisp	fuzzy
Type 5	crisp	crisp	fuzzy
Type 6	fuzzy	crisp	crisp
Type 7	fuzzy	fuzzy	crisp

Fuzzy NNs (FNN) of Type 1 are used in classification problem of a fuzzy input vector to a crisp class [189,190]. The networks of Type 2, 3 and 4 are used to implement fuzzy IF-THEN rules. However, the last three types in Table 1 are unrealistic. In Type 5, outputs are always real numbers because both inputs and weights are real numbers. In Type 6 and 7, the fuzzification of weights is not necessary because targets are real numbers.

5. VHDL Design of The Neuron

It is important to design the neuron without activation function as common part in designing a complete neuron with any activation function based on FPGA[4][8]. The design affects the utilization ratio of the chips area and the processing speed directly. The structure of the neuron can be realized in many ways, mainly considering the degree of the parallel computation needed.The proposed VHDL structural diagram for hardware implementation of neuron is shown in figure 6. The structure contains two shift registers , one shifters hold the weights , while the other holds the inputs (shift register with data load capability) .This approach is appropriate for general purpose neuron (i.e., with programmable weights). It employs only one input to load all weights (thus saving on chip pins). The weights are shifted in sequentially until the register is loaded with its weight. The weights are then multiplied by the input and accumulated to produce the desired output[3].

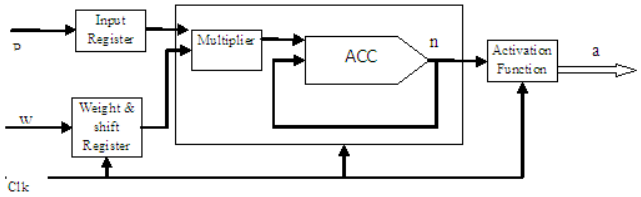

Figure 6: VHDL structural diagram for neuron implementation.

Simulation results are shown in figure 9. The neuron has three 4- bit input each. Since a SIGNED data

representation was employed, the range of the input values and weights runs from (-8 to 7) and the range of the 8-bit output runs from (-128 to 127) . The first input vector applied to the neuron has the values $p_1 = 3$, $p_2 = 4$, and $p_3 = 5$, since there are three weights, three clock cycles are needed to shift them in, as shown in figure 9 .The values chosen for the weights were $w_3 = 7$, $w_2 = 8$, $w_1 = 9$. Note that 9 is in indeed -7, and 8 is -8 because data type used here is SIGNED. Consequently, the weight have been all loaded, the system immediately gives its output, i.e., $a = p_1 w_1 + p_2 w_2 + p_3 w_3 = (3)(-7)+(4)(-8)+(5)(7) = -18$ - represent as 256- 18 =238. The neuron output for the second input vector [6 8 2] is 36. (Figure 7 shows the RTL (register transfer level) hardware schematic circuit for implementing linear neuron.

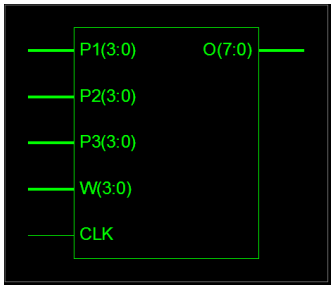

Figure 7: RTL of ANN

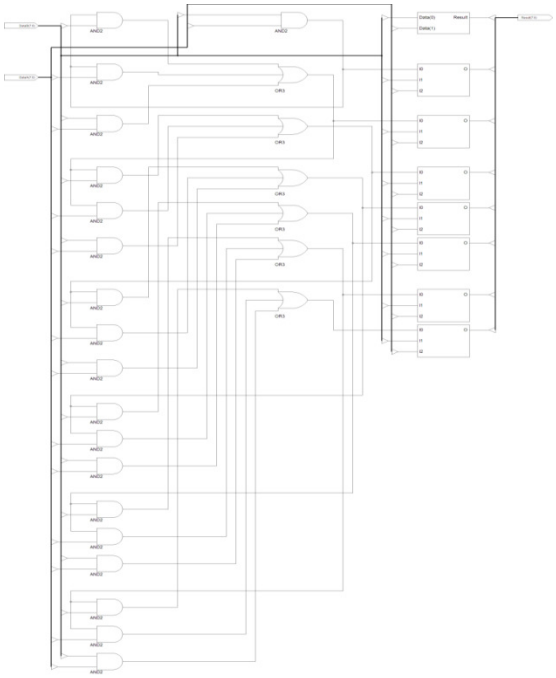

Figure 8: Internal RTL of multiplier-Adder.

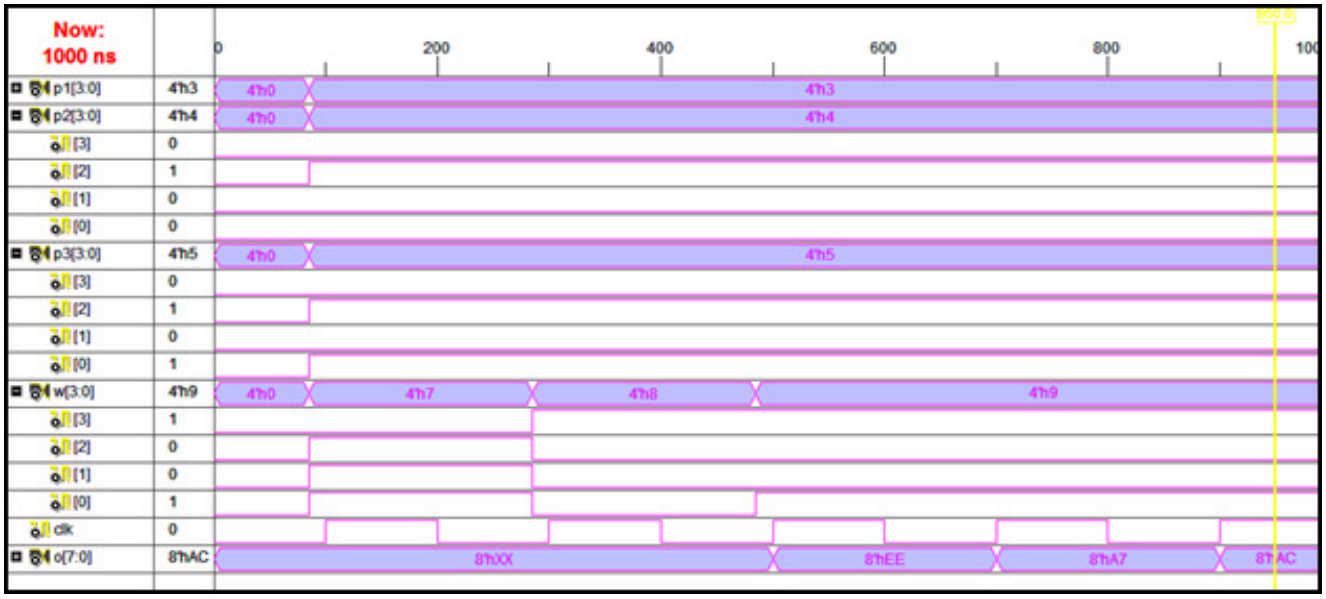

Figure 9 : Hardware circuit, for implementing an 8-bit linear artificial neuron (without activation function) Timing diagram.

6. Synthesis and Implementation Results

As a result of Synthesis and Implementation of artificial neuron with three different activation function (hardlims , satlins , tansig) on a Xilinx XC3S5OOE FPGA device .

Table 2 give performance and resource use summary for all implemented 8- bit neurons. The target device is xc3s50-5pqr208 and Software version 9.2i. As it can be seen , the hardlims function require a very few hardware

resource in comparison with the tansig function. The operation speed in all cases gives a good results and shows the advantages of using FPGAs in neural realization[8].

Table 2: Comparative data for the implemented artificial neurons

Neuron Type Device utilization	**Hardlims**	**satlins**	**Tansig**
Number of Slices (4656)	9	11	39
Number of Slices FF (9312)	8	8	8
Number of 4 input LUT's (9312)	14	21	72
Number of bonded IOB's (232)	25	25	25
Number of Multiplier (20)	3	3	4
Number of GCLK's (24)	1	1	1
Time Summary			
Maximum Path Delay	16.00ns	16.76ns	35.89ns
Maximum operating frequency	65MHz	59MHz	27.5MHz

7. Conclusion

The results of this work successfully demonstrate the hardware implementation of artificial neuron with three different activation functions using Xilinx FPGA's. This allows comparisons to be made between the hardware realizations of these neurons, which are regarded as basic building block of artificial neural networks. The tansig function approximation problem has been employed and demonstrates the validity of hardware implementation. The timing diagrams show the accuracy of the results and to enable comparisons with actual result. The operation frequency in all cases is very good and it gives a clear idea of the advantages of using FPGAs, since multiple modules can be working in parallel with a minimum reduction in performance due to the increased number of interconnections. Finally , it can be say that FPGAs technology and their low cost, and reprogrammability make this approach a very powerful option for implementing ANN and FIS as an alternative to the development of custom hardware.

References

[1] H. Okada, N. Watanabe, A. Kawamura and K. Asakawa, Initializing multilayer neural networks with fuzzy logic. in: Proceedings of the [2] K.L. Kar at. el., "VLSI implementation of OFDM" published in ELSEVIER , Volume 4, Issue 3, May 2012, ISBN: 978-93-5107-194-5 589-592.

[3] R. Kuc and V.B. Viard. "A Physically Based Navigation Strategy for Sonar Guided Vehicles," International Journal of Robotics, vol. 10, pp. 75-87, April 1991.

[4] K.L. Kar et. al.," Implementation of a fuzzy logic based automatic vehicle control system using VHDL." in International Journal of Advances in Science and Technology(IJAST) (Press).

[5] J.J .Buckley and Y. Hayashi, Neural nets for fuzzy systems, Fuzzy Sets and Systems, 71(1995) 265-276.

[6] K.L. Kar at. el., "Optimization of the performance of ANN using VHDL" published in International Journal of Scientific & Engineering Research, Volume 4, Issue 5, May 2013,ISSN 2229-5518.

[7] M.Hagan , H . Demuth , M. Beele , " Neural Network Design" , University of Colorado Bookstore, 2002, ISBN : 0- 9717321- 0-8.

[8] Parasovic A., latinovic I . , "A Neural Network FPGA Implementation" . IEEE. 5th seminar on Neural Network Application in Electrical Engineering , September 2000 , PP117 – 120.

Authors Profile

Dr. Shamita Chakraborty has 28.6 years of research, teaching and industrial experience, she acquired her Ph. D Degree in Electronics Engineering from Maulana Abul kalam Ajad National Institute of technology, Bhopal (M.P.) in the year 1996.Bagged the young scientist 1989 for the best research paper in Engineering Section and also received the Best Teacher award in the year 2000. She has 21 Technical Papers and articles to her credit during her work span as lecturer, Reader, and currently as professor in various AICTE approved Technical institutions. As an excellent orator and anchor, she has been associated in organizing, convening and conducting many national Symposiums, Workshops, Seminars and Conferences. She has proved her versatile personality in the capacity of Faculty Advisor, Co-coordinator and Prof-in-Charge for ALUMNI Association, Quality Circle, center for Creativity and innovation, Teaching Methodology and Publication of Annual Magazine as well, along with the cultural activities.

Mr. K. L. Kar born in West Bengal, India. He received his B. Tech. degree in Electronics from N.I.T., Rourkela, India in 2003, M.E.(VLSI Design), from C.S.V.T.U., India in 2009. He joined as an Asso. Prof. in E&TC department, MMCT, Raipur and working as a research engineer in A.M.G., Toronto, Canada since 2008. Currently doing PhD from Jadavpur University, Kolkata, India. He is a member of ACEEE and IDES. His present

interests are VLSI Design, Neural Network, Fuzzy Logic, Programming with VHDL and FPGA.

Dr. B. B. Mangaraj received the BE degree from University College of Engineering, Burla, Orissa, India, in 1994 and the ME TeLE degree and Ph. D. (Engg.) degree from Jadavpur University, Kolkata, India, in 2003 and 2012 respectively. In 2006, he joined the Department of Electronics and Telecommunication Engineering, University College of Engineering, Burla, as a Lecturer. In 2011 he joined as a Reader again in the same department. His present interests are in Analysis, design, coding and optimization of wire antenna structures and micro-strip antennas starting from simple to complicated one. He is also taking interest to apply VLSI in RF Engineering. He has been honored with university medal for his ME