Development of Artificial Neural Network on Field Programmable Gate Array

Jayu Kalambe, Richa Khandelwal and Meghana Hasamnis

***Abstract*-- This paper is concern with "Development of Artificial Neural Network on Field Programmable Gate Array". One of the characteristics of many industrial processes is the complex interrelation among the variables of the process. This has lead to the development of sensors, where by means of a computer program variables are estimated from the information gathered. On-line monitoring of all variables would be the best solution as off-line monitoring method means loss of information, delay in getting results and requires greater human efforts. Our work is based on a modular design of neural network using digital non-liner activation function. The ability of Artificial Neural Network to learn from experience rather from mechanistic description is making them the preferred choice to model processes with complex variables.**

This new sensor will be synthesized in a Field Programmable Gate Array to provide the process with the hardware version of the software sensor is simulated using ModelSim5.5c software and the simulation results have been presented in the paper.

***Index Terms*--** ANN, FPGA

I. INTRODUCTION

In case of hardware implementation of the ANN solution would be desirable, because it eliminates part of the cost and gives better level of confidence in the product, hence we thought for design of implementation of an ANN using Field programmable Gate Array (FPGA). Within the monitoring and control task required to optimize process operation , on-line monitoring of all variables would be the best solution , as off-line methods mean loss of information, delay in getting results and normally requires greater human effort., although on-line measurements would be desirable.

The network required for this purpose is multilayered network among which, we decided for implementing a single neuron. This developed neuron can be recursively used for obtaining the multilayer network. This multilayer network can be termed as a sensor, which can use for estimating the complex variables of the process which is going on in the industry.[4]

Jayu Kalambe, Lecturer,SRKNEC,Nagpur
Jayu_kalambe@rediffmail.com
Richa khandelwal Lecturer,SRKNEC,Nagpur
reechareema@rediffmail.com
Meghana Hasamnis Lecturer,SRKNEC,Nagpur
meghanahasamnis@rediffmail.com

The network required for this purpose is multilayered network among which, We have implemented a single neuron. This developed neuron can be recursively used for obtaining the multilayer network. This multilayer network can be termed as a sensor, which can use for estimating the complex variables of the process which is going on in the industry. Conventional computer hardware is not optimized for neural network processing and, while their implicit functions are comparatively simple, their hardware implementation can be expensive.[6]

II. HARDWARE AND IMPLEMENTATION OF ANN

The following expression is to be implemented in hardware,[1]

$$R_i(I_1,\ldots,I_m) = a_i\left(\sum_{1\le j\le m} w_{ji} * I_j\right)$$

where, Ii are the input signals, Ri is the set of input vectors, Wji the weight an a the activation.

Our implementation is based on a two input non-linear processing element (PE) with a digital sigmoid activation function. The design and all our work are geared towards the implementation of neuron in a modular fashion our modular design means that the network can become as large as practically possible thus providing a structure for complex application. The architecture and control logic were firstly defined in block diagram for further detailing. The software platform for the FPGA hardware implementation allows several levels of definition of logic function. One level is through hardware description languages (HDL). The block diagram of complete neuron is shown on figure 1 The circuit does the algebraic equations of the mathematical model of the neuron, that is, the multiplication and sum required in the neuron's internal processing. In this case, two inputs are specified and each is seventeen bits wide. Floating point binary representation (IEEE 754 standard) is used in order to be able to handle positive as well as negative data.

Figure 2 is given for multilayer neural network which consists of two hidden neuron and one output neuron, which requires six multipliers and three adders for its complete implementation. Here Floating-point Multiplier and Floating-Point Adder are used, as the inputs to them are in floating-point format. The two inputs are provided to the multiplexer.

The Multiplexer will select the input depending upon the select line and the first weight depending upon the address which is stored in ROM module as an input and will produce the output.[5] Then the output of the multiplexer is fed as an input to the multiplier where the inputs will get multiplied and the obtained result is then stored in the register bank_0. Then the next multiplexer will select the output of register as first data and the weight fro ROM module depending upon the select lines. Then the output of the multiplexer are taken as an input by the adder. The Adder will just add these two inputs. Then the output is passed through the non linear activation.

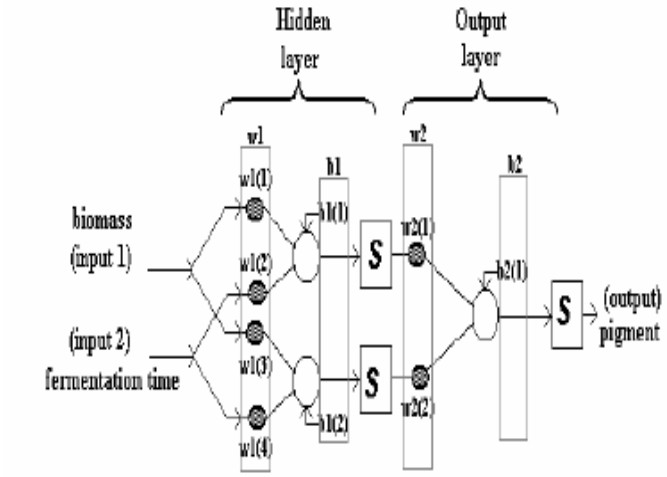

Figure 1 Block diagram of the complete neural network

III. REGISTER LEVEL ARTIFICIAL NEURAL NETWORK

IV. MODULES USED FOR DEVELOPMENT OF ANN

The whole software consists of several modules. Here the each input and output is 17 bits.

Floating point representation consists of sign, exponent and mantissa. Among 17 bit, 1 bit is sign, 6 bits are exponent and 10 bits are mantissa. There are in all eight modules control unit, floating-point adder module, floating-point multiplier module, rom module, register modules and multiplexer modules. These modules can be divided into asynchronous and synchronous. All the modules are first designed individually and then they are called in together to develop one neuron.

A. Control Unit

Control unit is designed to provide control signals to all the modules of a neuron. The clock and reset signal is taken as a input signals and the controlling signals of each modules has been taken as an output signals. The control unit was design using state machine mechanism. The input and output consists of 17 bits each. It is implemented in VHDL language. This module has the time control of all the modules that comprise the neuron. It is connected to ROM module.

.

B. Ram Modules

ROM module has been designed to provide weights to the respective inputs. In ROM the constants floating-point values are stored and these are read when needed,where the data are stored in floating point format. The data is nothing but the respective weights. These weights are multiplied by their respective inputs before summation. Here the inputs and weights both are 17 bits and hence we require 17 bit multiplier and 17 bit adder. These weights stored are

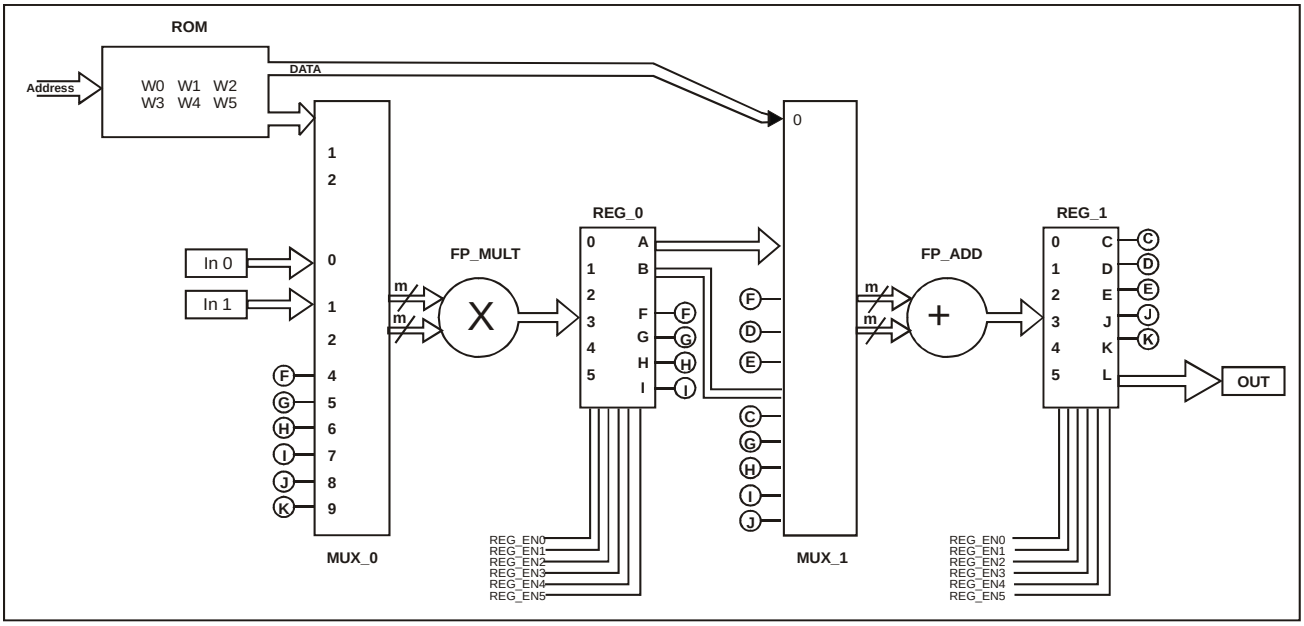

Figure 2. Functional block diagram of the complete neural network

constants values. It is connected to Control unit and multiplexer module

C. Register Banks

Registers are designed with "D" type flip-flop. Registers has been designed to store the calculated value after multiplication and addition. We require two register bank (reg_0 and reg_1), reg_0 for storing the output of multiplier and reg_1 for storing the output of adder. These register bank consists of six registers each of 17 bit. All this registers of register banks consists of 17 bit each hence we first design one 17 bit register and this has been called in each register bank six times. They are connected to multiplier and adder.

D. Multiplexers

Multiplexer modules are designed to select the multiple inputs. In all we require two multiplexers (Mux_0 and Mux_1), Mux_0 for selecting inputs from provided inputs and from ROM and Mux_1 for selecting inputs from ROM and output from reg_0. It is used for selecting weights and inputs. The multiplexer is also 17 bit as the inputs to the multiplexer is 17 bit

E. Floating-Point Multiplier

Floating-point multiplier is designed to multiply the inputs with their respective weights. There are two inputs which we are providing to the multiplexer, the multiplexer will select the desired input and the desired weight from the ROM module and will select the desired input and the weight. The inputs to the multiplier are multiplexer output and ROM output, multiplier will multiply these data. As the inputs to the multiplier are floating-point 17 bit data, hence we need to design 17 bit floating-point multiplier. The output of the multiplier is stored in the register bank. Architecture of Floating-point Multiplier is shown is figure below

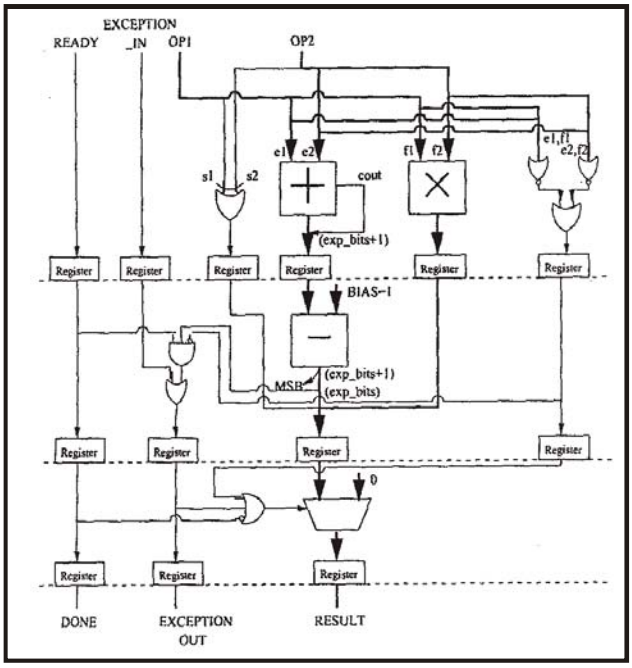

Figure 3. Floating-point Multiplier

In floating-point arithmetic, multiplication is a relatively straight-forward operation compared to addition. The fields of floating-point sign, mantissa and exponent do not interact during multiplication operation and can be thus processed at the same time, in parallel.

F. Floating-Point Adder

Floating-point adder is designed to add the multiplexer data and the ROM data. The ROM data is nothing but the weights of the respective inputs and multiplexer data, which is stored in register after multiplication. The inputs to the floating-point adder are from ROM output and multiplexer output. As the inputs to the adder are 17 bit hence we need to design 17 bit floating-point adder. The outputs of the floating-point adder are stored in the register bank. [2]

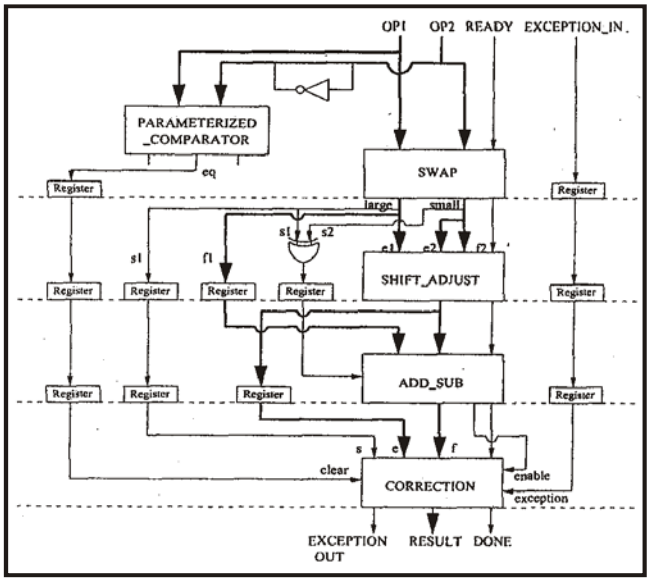

Figure 4: Floating-point Adder

Addition is one of the most computationally complex operations in floating-point arithmetic.

The output of the adder is passed through the sigmoid activation function which is the output of the software neuron. All these modules were individually design, and were complied and simulated which gives the software design of the neuron. These same weights are recorded in the ROM memory module of the neuron in hardware and are tested until output is obtained. Thus the software neuron is implemented on FPGA to obtain the hardware version of the software version.

V. SIMULATION RESUTLS

The neuron is the heart of all the neural network calculation i.e. multiplication of synaptic weights and their respective inputs components. [7]

The waveform shows two inputs input_0 and input_1. The input input_0 and ROM data of fist address is selected by the multiplexer_0 and

this two output of multiplexer_0 and given as an input to the multiplier where this two inputs are multiplied and the output is stored in the first memory location of the register bank_0.

Then the second input_1 and the next weight is selected and they are multiplied, which is stored in the second memory location of the register bank_0. Then these two outputs are added by the adder and are stored in the register bank_1. The output is then applied to the nonlinear activation function. Here we are using sigmoid as an activation function in its Taylor series form.

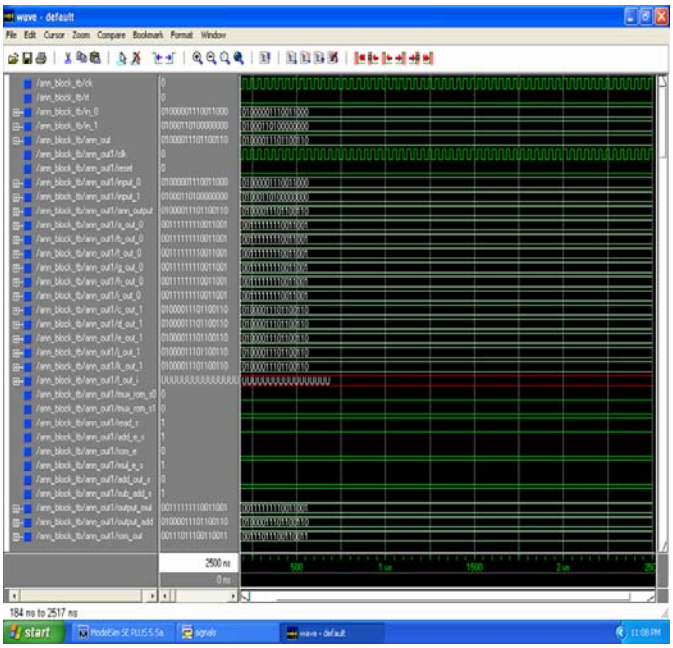

Figure 5 .Simulation Result for single Neuron

VI. CONCLUSION

The modeling of single neuron can be recursively used to implement a multilayer neural network. The new network obtained can be further used in industrial processes. The neural network implementation on Field Programmable Gate Array will give us the hardware implementation of the sensor. The hardware model obtained can be used in the on-line industrial processes.

The current design implements a two layer structure, future modification can have increase in number of neurons in hidden layer between the input an output layers which will improve the performance of network.

The addition of hidden layer not only finely tunes the network but also converges the weight value.

A. Units

Metric units are preferred for use in IEEE publications in light of their global readership and the inherent convenience of these units in many fields. In particular, the use of the International System of Units (Systeme Internationale d'Unites or SI Units) is advocated. This system includes a subsystem of units based on the meter, kilogram, second, and ampere (MKSA). British units may be used as secondary units (in parentheses). An exception is when British units are used as identifiers in trade, such as 3.5-inch disk drive.

B. Abbreviations and Acronyms

Define less common abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title unless they are unavoidable.

See Appendix A of the Author's Kit for additional information and standard abbreviations.

C. Math and Equations

Use either the Microsoft Equation Editor or the *MathType* commercial add-on for MS Word for all math objects in your paper (Insert | Object | Create New | Microsoft Equation *or* MathType Equation). "Float over text" should *not* be selected.

To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Use parentheses to avoid ambiguities in denominators.

Number equations consecutively with equation numbers in parentheses flush with the right margin, as in (1). Be sure that the symbols in your equation have been defined before the equation appears or immediately following.

$$I_F = I_B = -I_C = A^2 I_{A1} + A I_{A2} + I_{A0} = \frac{-J\sqrt{3}E_A}{Z_1 + Z_2} \qquad (1)$$

where I_F is the fault current.

Use "(1)," not "Eq. (1)" or "equation (1)," except at the beginning of a sentence: "Equation (1) is …."

VII. REFERENCES

[1].Marco A. Arroyo Leon, Arnoldo Ruiz Castro and Raul R. Leal Ascencio " An artificial neural network on a field programmable gate array as a virtual sensor". Jalisco , 45090,MEXICO.

[2].Pavle Belanovic "Library of Parameterized Hardware Modules for Floating-Point Arithmetic with An Example Application" Boston, Massachusetts, May 2002.

[3].Galindo Hernandez Miriam L., Leal Ascencio R.R. and Aguilera Galicia Cuauhtemoc "An Artificial Neural Network on a Complex Programmable Logic Device as a Virtual Sensor ", Technical Report, DESI, ITESO, Guadalajara, Mexico, January 1998.

[4].Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka "Elements of Artificial Neural Networks".

[5].Haykin, S..Neural Networks: "A Comprehensive Foundation".IEEE Press, Macmillan, New York, 1994.

[6].Lippmann, R.A. "An introduction to computing with neural nets". IEEE ASSP Magazine, page 4-21, 1987.

[7].Stephen Brown, Zvonko Vranesic "Fundamentals of Digital Logic with VHDL ."

VIII. BIOGRAPHIES

Ms. Jayu Kalambe, born in Katol in India on November 17, 1977. She graduated from the Priyadarshini College of Engineering, Nagpur (Maharashtra State) and studied her post graduation in Electronics Engineering from VNIT, Nagpur (Maharashtra State).

Her employment experience includes Seven years of teaching at graduate level. She is having to her credit many International and National Conference papers. Her special fields of interest include Neural Network and embedded system

Ms. Richa Khandelwal, born in Gwalior in India on June 05, 1976. She graduated from Madhav Institute of Technology and Science, Gwalior (Madhya Pradesh) and studied her post graduation in Electronics Engineering from Yeshwantrao Chavan College of Engineering, Nagpur , (Maharashtra State).

Her employment experience includes Seven years of teaching at graduate level. She is having to her credit many International and National Conference papers . Her special fields of interest include communication system and Embedded System