RESEARCH ARTICLE OPEN ACCESS

VLSI Architecture for Neural Network

Dhananjay Kumar[1], Dileep Kumar[2], J.R.Shinde[3], Amit Kumar[4], Vineet Kumar[5]
[1](Department of Electronics Engineering, RTM University Nagpur, INDIA
Email: dkraj88@gmail.com)
[2] (Department of Electronics Engineering, RTM University Nagpur, INDIA
Email: dileepjaiswal41@gmail.com)

ABSTRACT
In this paper a hardware implementation of an artificial neural network on Field Programmable Gate Arrays (FPGA) is presented. For the neural network based instrument prototype in real time application, conventional specific VLSI neural chip design suffers the limitation in time and cost. With the low precision artificial neural network (ANN) design, FPGAs have higher speed and smaller size for real time application than the VLSI design. In addition, artificial neural network (ANN) based on FPGAs has fairly achieved with classification application. The programmability of reconfigurable FPGAs yields the availability of fast special purpose hardware for wide applications. Its programmability could set the conditions to explore new neural network algorithms and problems of a scale that would not be feasible with conventional processor. The goal of this work is to design the optimal hardware for neural network (using FPGA). Neural Network system architecture designing using Very High Speed Integrated Circuits Hardware Description Language (VHDL) and farther transfer to FPGA board thus will provide an optimal solution for implementation of Neural Network system.

***Keywords* -** Artificial neural networks (ANNs), Field Programmable Gate Array (FPGA), Neuron Architecture, Sigmoid Activation Function, Very High Speed Integrated Circuits Hardware Description Language (VHDL).

I. INTRODUCTION

Artificial neural networks (ANN) have found widespread deployment in a broad spectrum of classification, perception, association and control applications [1].

The advance in high speed computing has proved through simulation the capability of Artificial Neural Networks (ANN) to map and classify nonlinear systems. For the Real time applications are possible only if low cost high-speed neural computation is made realizable. Towards this goal numerous works on implementation of Neural Networks (NN) have been proposed [2].

Artificial Neural Networks have been mostly implemented in software; this has been benefits, since the designer does not need to know the inner working in neural network elements, Most of the work done in this field until now consists of software simulations, investigating capabilities of ANN models or new algorithms. But hardware implementations are also essential for applicability and for taking the advantage of neural network’s inherent parallelism.

There are analog, digital and also mixed system architectures proposed for the implementation of ANNs. The analog ones are more precise but difficult to implement and have problems with weight storage. Digital designs have the advantage of low noise sensitivity, and weight storage is not a problem. With the advance in programmable logic device technologies, FPGAs has gained much interest in digital system design. They are user configurable and there are powerful tools for design entry, syntheses and programming.

ANNs are biologically inspired and require parallel computations in their nature. Microprocessors and DSPs are not suitable for parallel designs. Designing fully parallel modules can be available by ASICs and VLSIs but it is expensive and time consuming to develop such chips. In addition the design results in an ANN suited only for one target application. FPGAs not only offer parallelism but also flexible designs, savings in cost and design cycle.

II. ARTIFICIAL NEURON

Artificial neural networks are inspired by the biological neural systems. The transmission of signals in biological neurons through synapses is a complex chemical process in which specific transmitter substances are released from the sending side of the synapse. The effect is to raise or lower the electrical potential inside the body of the receiving cell. If this potential reaches a threshold, the neuron fires. It is this characteristic of the biological neurons that the artificial neuron model proposed by McCulloch Pitts attempts to reproduce. Following neuron model shown in Fig. 1 is widely used in artificial neural networks with some variations.

The artificial neuron given in this figure has N inputs, denoted as p_1, p_2...p_N. Each line connecting these inputs to the neuron is assigned a weight, denoted as w_1, w_2,…, w_N respectively. The activation, a, determines whether the neuron is to be fired or not. It is given by the formula.

$$a = \sum_{j-1}^{N} W_j P_j \tag{1}$$

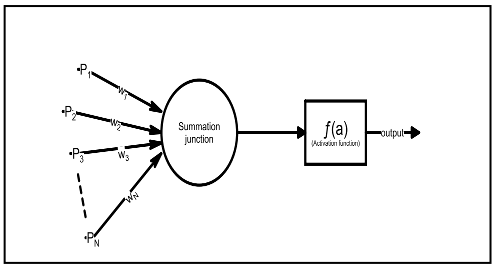

Fig.1. Structural diagram of a neuron

A negative value for weight indicates an inhibitory connection while a positive value indicates excitatory connection.

The output, Y of the neuron is given as:

$$Y= f(a) \tag{2}$$

Originally the neuron output function f(a) in McCulloch Pitts model was proposed as threshold function, however linear, ramp, and sigmoid functions are also used in different situation. The vector notation

$$a = w^T p \tag{3}$$

it can be used for expressing the activation of a neuron. Here, the *jth* element of the weight vector of *w* is *wj*. Both of these vectors are of size *N*. A Neuron-computing system is made up of artificial neurons and a huge number of interconnections between them.Fig.2 show architecture of feed-forward neural network [5].

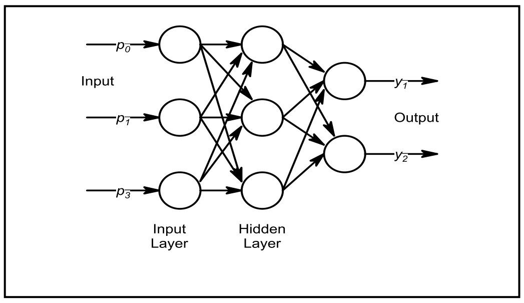

Fig.2. Layered feed-forward neural network

In layered neural network, the neurons are organized in the form of layers. The neurons in a layer get inputs from the previous layer and feed their output to the next layer. These types of networks are called feed-forward networks.

Output connections from a neuron to the same or previous layer neurons are not permitted. The input layer is made of special input neurons, transmitting only the applied external input to their outputs. In a network, if there are input and output layers only, then it is a single layer network. Networks with one or more hidden layers are called multilayer networks [6].

III. OVERVIEW OF VHDL

VHDL is language meant for describing digital electronics system. In its simplest form, the description of component in VHDL consists of an interface specification and an architectural specification. The interface description begins with the ENTITY keyword and contains the input- output ports of the component. The name of the component comes after the ENTITY keyword and is followed by IS, which is also a VHDL keyword. The description of the internal implementation of an entity is called an architecture body of the entity. There may be number of different architecture bodies of an interface to an entity corresponding to alternative implementations that perform the same function.

After describing a digital system in VHDL, simulation of the VHDL code has to be carried out for two reasons. First, we need to verify whether the VHDL code correctly implements the intended design. Second, we need to verify that the design meets its specification. The simulation is used to test the VHDL code by writing test bench model. A test bench model that is employed to exercise and verify the correctness of a hardware model and it can be described in the same language.

Some synthesis tools are capable of implementing the digital system described by VHDL code using a

PGA (programmable gate array) or CPLD (complex programmable logic devices), they are more flexible and more versatile and can be used to implement a complete digital system on a single chip. The user can program the functions realized by each logic cell and the connections between the cells. Such PGAs are often called FPGAs (field programmable logic array) [7], [8].

IV. PROPOSED DESIGN

The proposed design is VLSI architecture for neural network. It Consist of neurons architecture design, activation function problem solving and artificial NN design which consist of two layers.

4.1 Neurons Architecture

The processing element of an ANN is Neuron. A Neuron can be viewed as processing data in three steps, the weighting of its input values, the summation of them all and their filtering by sigmoid function. The summation can be calculated by a serial accumulation. For the weighted input to be calculated in parallel using conventional design techniques, a large number of multiplier units would be required. To avoid this, Multiplier/Accumulator architecture has been selected. It takes the input serially, multiplies them with the corresponding weight and accumulates their sum in register. The processes are synchronized to clock signal. The number of clock for a neuron to finish its work, equals to the number of connections from the previous layer. The accumulator has a load signal, so that the bias values are loaded to all neurons at start-up. Fig. 3 shows the proposed neuron design.

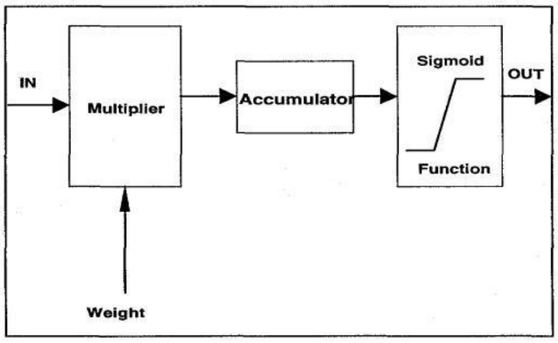

Fig. 3 Neuron architecture

4.2 Activation Function

The most important parts of a neuron is its activation function. The nonlinearity of the activation function makes it possible to approximate any function. In the hardware implementation concept of neural networks, it is not so easy to realize sigmoid activation functions [9].

Special attention must be paid to an area-efficient implementation of every computational element when implementing large ANNs on digital hardware. This holds true for the nonlinear activation function used at the output of neurons.

A common activation function is the sigmoid function

$$Y = \frac{1}{1+\varepsilon^{-P}} \tag{4}$$

Efficient implementation of the sigmoid function on an FPGA is a difficult challenge faced by designers. It is not suitable for direct implementation because it consists of an infinite exponential series. In most cases computationally simplified alternatives of sigmoid function are used.

Direct implementation for non-linear sigmoid transfer functions is very expensive. There are two practical approaches to approximate sigmoid functions with simple FPGA designs. Piece-wise linear approximation describes a combination of lines in the form of y= (ap + b) which is used to approximate the sigmoid function. Especially if the coefficients for the lines are chosen to be powers of two, the sigmoid functions can be realized by a series of shift and add operations. The second method is lookup tables, in which uniform samples taken from the centre of a sigmoid function can be stored in a table for look up. The regions outside the centre of the sigmoid function are still approximated in a piece-wise linear fashion.

This research presents an approximation approach to implement sigmoid function. A simple second order nonlinear function can be used as an approximation to a sigmoid function. The structural diagram of the approximated sigmoid function implementation is shown in Fig. 4.

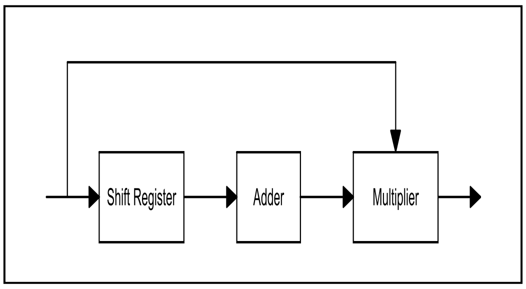

Fig. 4 structural diagram for sigmoid function

4.3 Layer Architecture

In fist Implementation of an ANN layer they take an input from their common input line, multiply it with the corresponding weight from their weight ROM and accumulate the product. If the previous layer has 3 neurons, present layer takes and processes these inputs in 3 clock cycles. After these 3 clock cycles, every neuron in the layer has its net values ready. Then the layer starts to transfer these values to its output one by one for the next layer to take them successively by enabling corresponding neuron's three-state output. The block diagram of a layer architecture including 3 neurons is shown in Fig. 5.

Since only one neuron's output have to be present at the layer's output at a time, instead of implementing an activation function for each neuron it is convenient to implement one activation function for each layer. In this layer structure pipelining is also possible. A new input pattern can enter the network while another is propagating through the layers.

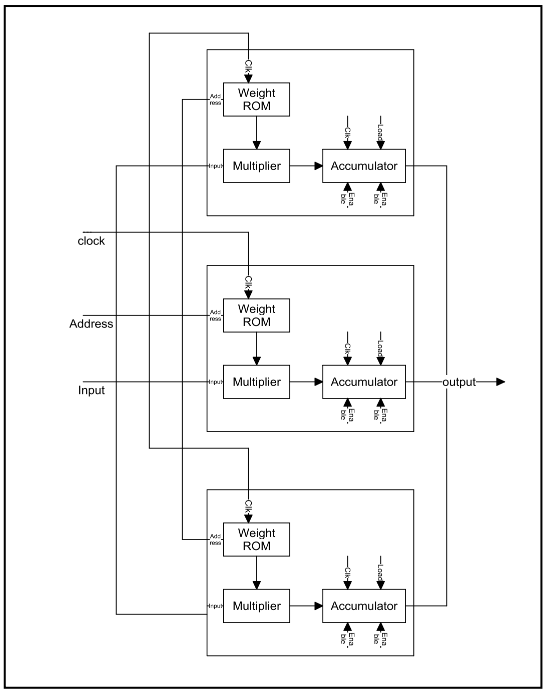

Fig.5 Block diagram of a layer consist of three neuron

A second Implementation of a fully parallel neural network is possible in FPGAs. A fully parallel network is fast but inflexible. Because, In a fully parallel network the number of multipliers per neuron must be equal to the number of connections to this neuron. Since all of the products must be summed, the number of full adders equals to the number of connections to the previous layer minus one. For example in a 3-1 network the output neuron must have 3 multipliers and 2 full adders. So, different neuron architectures have to be designed for each layer. Because the multipliers are most resource using elements in a neuron structure, a second drawback of a fully parallel network is gate resource usage. Fig. 6 show single layer artificial neural network with three input nodes and one output node.

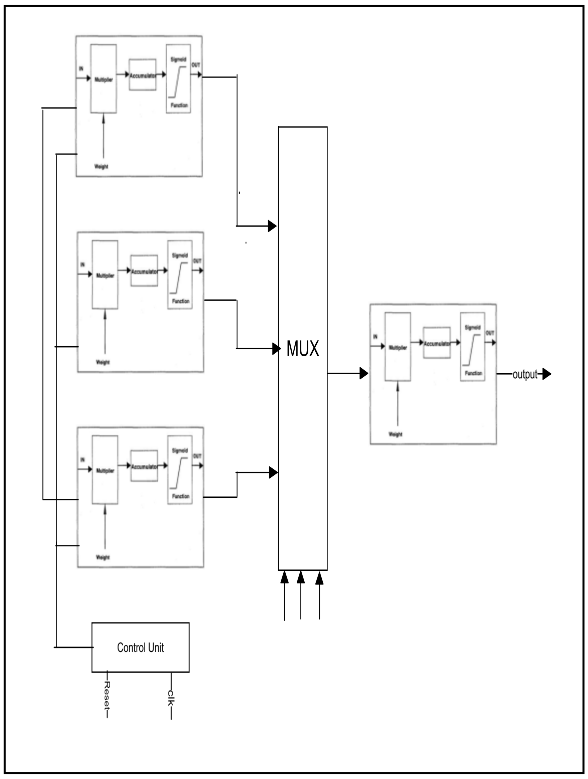

Fig. 6 Artificial neural network design

V. CONCLUSIONS

This paper has presented the idea of implementation of neural networks using FPGAs. The resultant neural networks are modular, compact, and efficient and the number of neurons, number of hidden layers and number of inputs are easily changed.

The VHDL-FPGA combination is shown to be a very powerful embedded system design tool, with low cost, reliability, and multi-faceted applications. As FPGAs allow the hardware design via configuration software control, the improvement of circuitry design is just a matter of modifying, debugging and downloading the new configuration code in a short time.

Future work involves estimating the maximum size of ANNs in modern FPGAs. The main points are the size and parameter stability of multipliers and the number of interlayer interconnections. The first defines mainly the required area resources and the second defines the required routing.

REFERENCES

[1] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: applications in industry, business and science," *Communications of the ACM*, vol. 37, no. 3, pp. 93-105, 1994.

[2] Muthuramalingam, S. Himavathi, and E. Srinivasan, "Neural network implementation using fpga: issues and application," *The International Journal of Information Technology*, vol. 4, no. 2, pp.86-92, 2008.

[3] M. T. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," *IEEE Proceedings, Computers and Digital Techniques, vol.* 150, no. 6, pp. 403- 411, 2003.

[4] Savran and S. Ünsal, "Hardware implementation of a feedforward neural network using FPGAs'," Ege University, Department of Electrical and Electronics Engineering, 2003.

[5] S. Haykin, *Neural Networks-a Comprehensive Foundations*, Second Edition, ISBN: 0132733501, 1998.

[6] S. Rai and A. P. Singh, "A review of implementation techniques for artificial neural networks," University School of Information Technology, GGS Indraprastha University, Delhi, 2006.

[7] S. P. J. V. Rani and P. Kanagasabapathy, "Design of Neural Network on FPGA," *International Conference on VLS*, USA, 2004.

[8] J. Hamblen and M. Furman, *Rapid prototyping of digital systems*, Kluwer Academic Publisher 2nd Edition, Boston, 2001.

[9] M. Avcýand T. Yýldýrým, "Generation of tangent hyperbolic sigmoid function for microcontroller based digital implementation of neural networks," in *Proc. International XII. Turkish Symposium on Artificial Intelligence and Neural Networks*, 2003.