Implementation Of A Radial Basis Function Using VHDL

GJCST Classification
I.2.6. K.3.2

[1]D.C. Dhubkarya , [2]Deepak Nagariya , [3]Richa Kapoor

***Abstract*- This paper presents the work regarding the implementation of neural network using radial basis function algorithm on very high speed integrated circuit hardware description language (VHDL). It is a digital implementation of neural network. Neural Network hardware has undergone rapid development during the last decade. Unlike the conventional von-Neumann architecture that is sequential in nature, Artificial Neural Networks (ANNs) Profit from massively parallel processing. A large variety of hardware has been designed to exploit the inherent parallelism of the neural network models.**

The radial basis function (RBF) network is a two-layer network whose output units form a linear combination of the basis function computed by the hidden unit & hidden unit function is a Gaussian. The radial basis function has a maximum of 1 when its input is 0. As the distance between weight vector and input decreases, the output increases. Thus, a radial basis neuron acts as a detector that produces 1 whenever the input is identical to its weight vector.

Keywords- RBF, training algorithm, weight, block RAM, FPGA.

I. INTRODUCTION

Artificial Neural Networks (ANNs) are non-linear mapping structures based on the function of the human brain. They are powerful tools for modeling, especially when the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are Imprecise and noisy. An ANN is a computational structure that is inspired by observed process in natural networks of biological neurons in the brain. It consists of simple computational units called neurons, which are highly interconnected. ANNs have become the focus of much attention, largely because of their wide range of applicability and the ease with which they can treat complicated problems. ANNs are parallel computational models comprised of densely interconnected adaptive processing units.[1] These networks are fine-grained parallel Implementations of nonlinear static or dynamic systems. A very important feature of these networks is their adaptive nature, where ―learning by example" replaces ―programming" in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily Available. ANNs are now being increasingly recognized in the area of classification and prediction, where regression model and other related statistical techniques have traditionally been employed.

[1]*D.C. Dhubkarya , UPTU, Lucknow , BIET Jhani, dcd3580@yahoo.com*
[2]*Deepak Nagariya , UPTU, Lucknow , BIET Jhansi , deepaknagaria@gmail.com*
[3]*Richa Kapoor, UPTU,Lucknow , SIT,Mathura, richakapoor11@gmail.com*

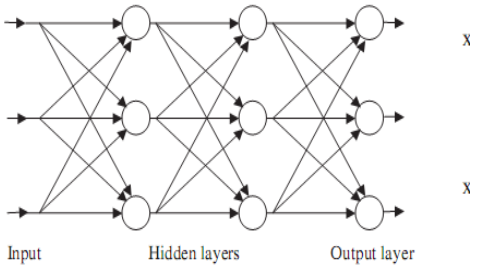

Fig. 1: Schematic representation of neural network

II. OVERVIEW OF RADIAL BASIS FUNCTION (RBF) NETWORKS"

Radial basis function (RBF) neural network consist of three layers, an input, a hidden and an output. It has a feed forward structure consisting of a single hidden layer of J locally tuned units, which are fully interconnected to an output layer of L linear units. All hidden units simultaneously receive the n-dimensional real-valued input vector X .The hidden-unit outputs are not calculated using the weighted-sum mechanism/sigmoid activation; rather each hidden-unit output $\varphi_i(X)$ is obtained by closeness of the input X to an n-dimensional parameter vector Cj associated with the jth hidden unit. [13] The response characteristics (activation function) of the jth hidden unit (j = 1, 2, .. J) is assumed as,

$$\varphi_i(X) = \exp\left\{-\frac{\|X - C_i\|^2}{2\sigma_i^2}\right\}.$$

The Parameter σj is the width of the receptive field in the input space from unit j. This implies that $\varphi_i(X)$ has an appreciable value only when the distance $\|X - C_i\|$ is smaller than the width σj..

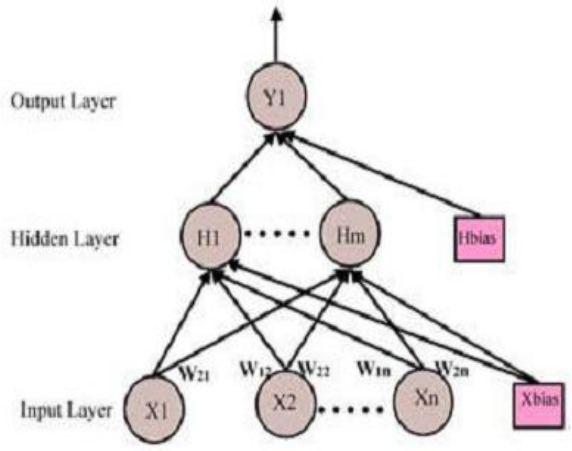

Fig2: Feed Forward Neural Network

RBF networks are best suited for approximating continuous or piecewise continuous real-valued mapping. $f: R^n \rightarrow R^L$, where n is sufficiently small. These approximation problems include classification problems as a special case. In the present work we have used a Gaussian basis function for the hidden units. RBF networks have been successfully applied to a large diversity of applications including interpolation, chaotic time-series modeling, system identification, control engineering, electronic device parameter modeling, channel equalization, speech recognition, image restoration, shape- from-shading, 3-D object modeling, motion estimation and moving object segmentation etc [7].

III. TRAINING OF RBF NEURAL NETWORKS

By means of training, the neural network models the underlying function of a certain mapping. In order to model such a mapping we have to find the network weights and topology. There are two categories of training algorithms: supervised and unsupervised. In supervised learning, the model defines the effect one set of observations, called inputs, has on another set of observations, called outputs. In other words, the inputs are assumed to be at the beginning and outputs at the end of the causal chain. The models can include mediating variables between the inputs and outputs. In unsupervised learning, all the observations are assumed to be caused by latent variables, that is, the observations are assumed to be at the end of the causal chain. In practice, models for supervised learning often leave the probability for inputs undefined.[1]

RBF networks are used mainly in supervised applications. In a supervised application, we are provided with a set of data samples called training set for which the corresponding network outputs are known.

RBF networks are trained by

i. deciding on how many hidden units there should be
ii. deciding on their centres and the sharpnesses (standard deviation) of their Gaussians
iii. Training up the output layer.

In training phase, a set of training instances is given. A feature vector typically describes each training instances. It is further associated with the desired outcome, which is further represented by a feature vector called output vector. Starting with some random weight setting, the neural net is trained to adapt itself by changing weight inside the network according to some learning algorithm. When the training phase is complete the weights are fixed. The network propagates the information from the input towards the output layer. When propagation stops, the output units carry the result of the inferences.

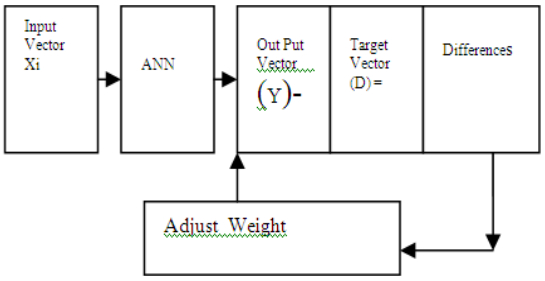

Fig 3: Block Diagram

Learning law describes the weight vector for the ith processing unit at time instant (t+1) in terms of the weight vector at time instant (t) as follows;

$$w_i(t+1) = w_i(t) +/- \Delta w_i(t),$$

Where $\Delta w_i(t)$ is the change in the weight vector. The Networks adapt change the weight by an amount proportional to the difference between the desired output and the actual output. As an equation:

$\Delta W_i = \eta * (D-Y).Xi$

Here E=D-Y

The perceptron learning rule can be written more succinctly in terms of the error E and the change to be made to the weight vector ΔW_i

CASE 1- If E = 0, then make a change ΔW equal to 0.

CASE 2- If E = +, then make a change

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

CASE 3-. If E = -1, then make a change

$$w_i(t+1) = w_i(t) - \Delta w_i(t),$$

Where η is the learning rate, D is the desired output, Y is the actual output, and Ii is the ith input. The weights in an ANN, similar to coefficients in a regression model, are adjusted to solve the problem presented to ANN. Learning or training is term used to describe process of finding values of these weights. Two types of learning with ANN are supervised and unsupervised learning. An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error.

IV. IMPLEMENTATION

Parameter

η=0.8;

Weight vector=8;

Input Vector=8;
Target value=1;

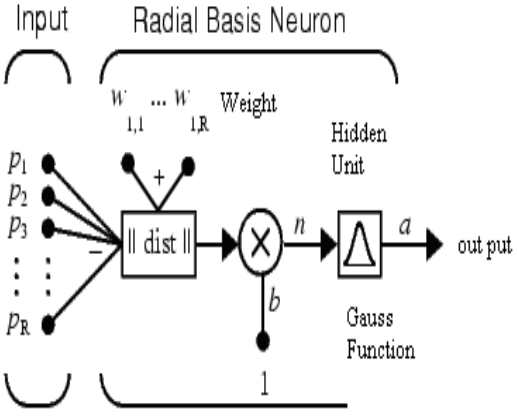

Fig: 4 Neuron Model

$f(\vec{x}) = e^{-Cu^2}$ where C is a constant and $u = \sqrt{\sum_i ((x_i - \omega_i)^2)}$

Since in the transfer function the u is squared, the square-root in u is unnecessary (especially in the hardware), and the function becomes

$$f(\vec{x}) = e^{-Cv} \quad , \quad v = \sum_i ((x_i - \omega_i)^2)$$

(The `|| dist ||` box in this figure accepts the input vector **p** and the single row input weight matrix.

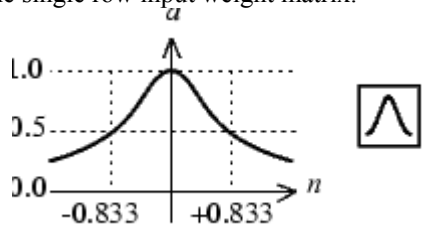

Fig 5: Radial Basis Function

we can understand how this network behaves by following an input vector **p** through the network to the output **a**. we present an input vector to such a network, each neuron in the radial basis layer will output a value according to how close the input vector is to each neuron's weight vector. Thus, radial basis neurons with weight vectors quite different from the input vector **p** have outputs near zero. These small outputs have only a negligible effect on the linear output neurons.

In contrast, a radial basis neuron with a weight vector close to the input vector **p** produces a value near 1. If a neuron has an output of 1, its output weights in the second layer pass their values to the linear neurons in the second layer.

In fact, if only one radial basis neuron had an output of 1, and all others had outputs of 0's (or very close to 0), the output of the linear layer would be the active neuron's output weights. This would, however, be an extreme case. Typically several neurons are always firing, to varying degrees. if it's output is not 1 then weight is adjusted according to training algorithm used & weight is updated till desired valued matched with target value.

V. Simulation Results

The proposed design was coded in VHDL. It was functionally verified by writing a test bench and simulating it using ISE simulator and synthesizing it on Spartan 3A using Xilinx ISE 9.2i.

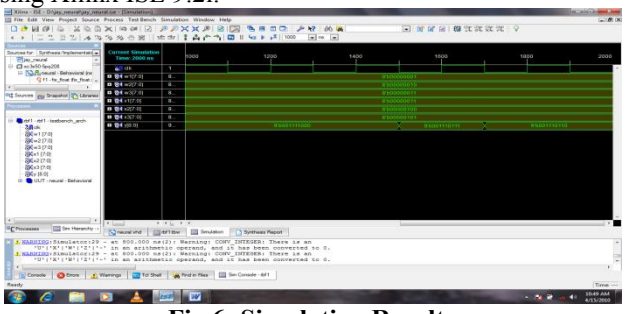

Fig 6: Simulation Results

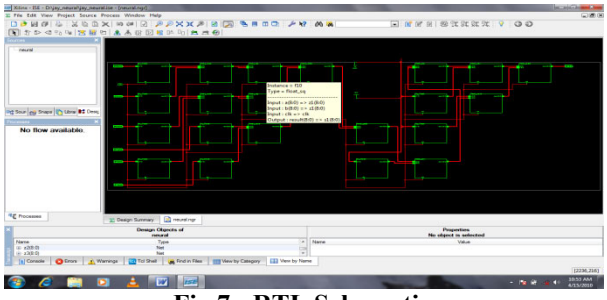

Fig 7 : RTL Schematic

HDL Synthesis Report

Output File Name —neural"
Output Format: NGC
Target Device: xc3s50-5-pq208
Number of Slices: 1000 out of 768
Number of Slice F/F: 211 out of 1536
Number of 4 input LUTs: 1713 out of 1536
Number of IOs: 58
Number of bonded IOBs: 58 out of 124
Number of GCLKs: 1 out of 8 12%

VI. References

1) Adrian G. Bors —Introduction of the Radial Basis Function (RBF) Networks" Department of Computer Science University of York YO10 5DD, UK.
2) Broomhead, D. S., Jones, R., McWhirter, J. G., Shepherd, T. J., (1990) —Systolic array for nonlinear multidimensional interpolation using radial basis functions," Electronics Letters, vol. 26, no. 1, pp. 7-
3) Bors, A.G., Pitas, I., (1996) —Median radial basis functions neural network," IEEE Trans. on Neural Networks, vol. 7, no. 6, pp. 1351-1364.
4) Casdagli, M. (1989) Nonlinear prediction of chaotic time series," Physica D, vol. 35, pp. 335-356.

5) Chen, S., Cowan, C. F. N., Grant, P. M. (1991) —Orthogonal least squares learning Algorithm for radial basis function networks," IEEE Trans. On Neural Networks, vol. no. 2, pp. 302-309.
6) Douglas L Perry (2006), —VHDL Programming by Example", McGraw- Hill.
7) F. Belloir, A. Fache and A. Billat (1998), —A New Construction Algorithm of efficient Radial Basis Function Neural Net Classifier and its Application to codes Identification",Ardenne, B.P. 1039, 51687.
8) Igelnik, B., Y.-H. Pao, (1995) —Stochastic choice of radial basis functions in adaptive function approximation and the functional-link net," IEEE Trans. on Neural Networks, vol. 6, no. 6, pp. 1320-1329.
9) Karayiannis,N.B.(1999) Reformulated radial basis neural networks trained by gradient descent," IEEE Trans. on Neural Networks, vol. 10, no. 3, pp. 657-671.
10) Kohonen,T.K.,(1989)Self-organization and associative memory. Berlin: Springer- Verlag.
11) Karen Parnell & Nick Mehta (2002), —Programmable Logic Design Quick Start Hand Book", Xilinx.
12) Musavi, M.T., Ahmed, W., Chan, K.H., Faris,K.B., Hummels, D.M., (1992) —On the training of radial basis function classifiers," Neural Networks, vol. 5, pp. 595-603.
13) P. Venkatesan* and S. Anitha,—Application of a radial basis function neural network for diagnosis of diabetes mellitus" Tuberculosis Research Centre, ICMR, Chennai 600 031, India
14) Satish Kumar —Neural Networks —A classroom approach, TMH.
15) Yuehui Chen1, Lizhi Peng1, and Ajith Abraham (2004)," Hierarchical Radial Basis Function NeuralNetworks for Classification Problems" International Journal of Neural Systems, 14(2):125-137.