FPGA Realization of a Radial Basis Function Based Nonlinear Channel Equalizer*

Poyueh Chen[1], Hungming Tsai[2], ChengJian Lin[3,**], and ChiYung Lee[3]

[1] Department of Computer Science and Information Engineering
[2] Institute of Networking and Communication Engineering
Chaoyang University of Technology
[3] Department of Computer Science and Information Engineering
Nan Kai Institute of Technology
168 Gifeng E. Rd., Wufeng Taichung County, 413 Taiwan, China
568 Chung Cheng Rd., Tsao Tun, 542, Nan Tou County, Taiwan, China
cjlin@mail.cyut.edu.tw

Abstract. In this paper we propose a radial basis function (RBF) neural network for nonlinear time-invariant channel equalizer. The RBF network model has a three-layer structure which is comprised of an input layer, a hidden layer and an output layer. The learning algorithm consists of unsupervised learning and supervised learning. The unsupervised learning mainly adjusts the weight among input layer and hidden layer. The supervised learning adjusts the weight among output layer and hidden layer. We will implement RBF by using FPGA. Computer simulation results show that the bit error rates of the RBF equalize using software and hardware implements are close to that of the optimal equalizer.

1 Introduction

During the past few years, applications of high-speed communication are required and fast increasing. Nonlinear distortion becomes a major fact or which limits the performance of a communication system. High speed communications channels are often impaired by the channel inter-symbol interference (ISI), the additive white Gaussian noise (AWGN), and co-channel interference (CCI) [1]. All these effects are nonlinear and complex problems. Nevertheless, adaptive equalizers are used in digital communication system receivers to mitigate the effects of non-ideal channel characteristics and to obtain reliable data transmission.

We will adopt the radial basis function (RBF) neural network [2] suggested by Moody and Darken. Radial basis function (RBF) neural networks provide an attractive alternative to MLP for adaptive channel equalization problems because the structure of the RBF network has a close relationship to Bayesian methods for channel equalization and interference rejection problems. The main benefit was using linear algebra's basis

* This research is supported by the National Science Council under grant NSC 92-2213-E-252-006.

** Corresponding author.

J. Wang, X. Liao, and Z. Yi (Eds.): ISNN 2005, LNCS 3498, pp. 320–325, 2005.

mathematic, relatively reduced the computation load. Usually the computation quantity problem takes much more time to simulate through software. The solution is the utilization of hardware simulation to obtain faster computational effectiveness.

Development of digital integrated circuit Field Programmable Gate Array (FPGA) [3]-[5] digitizes the hardware making process. Recently programmable logic element increases the number of the logic, velocity and memory. And add a lot of extra function. In addition, use Very High Speed Integrated Circuit Hardware Description Language (VHDL), enable the complicated circuit to form the way through the circuit that VHDL compile, reach the specification designed easily and fast. Hence it holds lots of benefits like high capacity, speedy, duplicate design, cheaper price, and cost lower. Finally we will achieve this adapted, radiation basis function neural network equalizer by using hardware FPGA.

2 The Structure of RBF

The structure of the RBF neural network model is shown in Fig.1. The input data in the input layer of the network is $x = [x_1, x_2, x_3, \ldots, x_n]$, where n is the number of dimensions. The hidden layer consists of *m* computing units (ϕ_1 *to* ϕ_m), which are connected to the output by m connection weights (w_1 *to* w_m).

The output of the network used by this algorithm has the following form:

$$Y(x) = f(x) = \sum_{j=1}^{m} w_j \phi_j(x) \tag{1}$$

where ϕ_j is the response of the *j*th hidden neuron to the input x, is the weight connecting the *j*th hidden unit to the output unit. Here, *m* represents the number of hidden neurons in the network, and ϕ_j is a Gaussian function given by

$$\phi_j = exp(-\frac{\|x - c_j\|^2}{\sigma_j^2}) \tag{2}$$

where c_j is the center and σ_j is the width of the Gaussian. $\|\|$ denotes the Euclidean norm.

3 The Learning Algorithm for RBF

The learning algorithm consists of unsupervised learning and supervised learning. The unsupervised learning mainly adjusts the weight among input layer and hidden layer. The supervised learning adjusts the weight among output layer and hidden layer.

3.1 Unsupervised Learning

The unsupervised k-means clustering procedure is often employed as a part of the general learning algorithm to adjust RBF centers. This involves computing the squared distance between the centers and the network input vector, selecting minimum squared

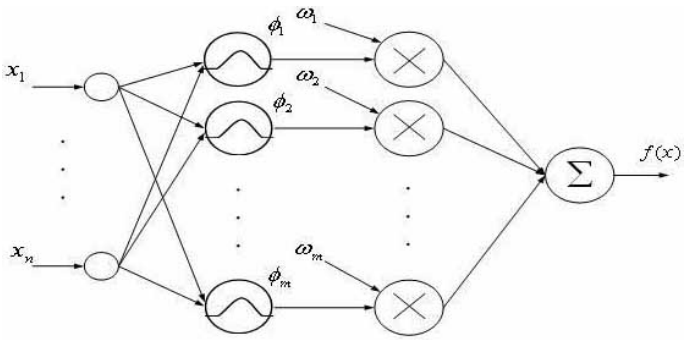

Fig. 1. RBF neural network structure.

distance and moving the corresponding center closer to the input vector. The computational procedure of this unsupervised clustering is as follows:

$$d_j(s) = \|x(s) - c_j(s-1)\|^2, \quad 1 \le j \le n \tag{3}$$

$$l = arg[min\{d_j(s), \quad 1 \le j \le n\}] \tag{4}$$

$$c_l(s) = c_l(s-1) + \alpha_c(x(s) - c_l(s-1)) \tag{5}$$

$$c_j(s) = c_j(s-1), \quad 1 \le j \le n \ \ and \ j \ne l \tag{6}$$

3.2 Supervised Learning

The supervised algorithm is very simple and robust. It is advisable to adjust the weights of the network so that the network can learn the general equalizer solution. The adaptation of the weights is achieved using the following supervised algorithm:

$$\phi_j(s) = exp(-\frac{\|x(s) - c_j(s)\|^2}{\sigma^2}), \quad 1 \le j \le n \tag{7}$$

$$\varepsilon(s) = t(s-\tau) - \sum_{j=1}^{n} w_j(s-1)\phi_j(s) \tag{8}$$

$$w_j(s) = w_j(s-1) + \alpha_w \varepsilon(s)\phi_j(s), \quad 1 \le j \le n \tag{9}$$

4 Hardware Implementation

We can plan FPGA into the function that a user wants through the design of CLB and connection of the connecting wire. As for we design FPGA, the assisting of software is needed. So we utilize ISE 6.2i software that Xilinx Company produces to do logic design.

The main part of the network structure of RBF neural network is design of hidden layer. The input layer can direct output of input value to input of hidden layer. The process does not pass through any operation. The output layer only needs to add up all

outputs which hide layer with the adder. The part of hidden layer is to consist of gaussian function. The structure of gaussian function is made up by a subtraction, divider, multiplier and exponential function. The component shows to Fig.2. The part of the multiplier includes the adder and counter. The concept implementation use bit shift and add up it. The exponential function part can rely on Taylor's expansion approximatively. Taylor's expansion is as follows:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^m}{m!} \tag{10}$$

where *m* is number of order, x is input. The larger *m* is the more perfect, but it will be many logic gates and bit numbers used. We choose *m*=4 to make the realization of the hardware. Fig.3 shows the block diagram that its hardware implementation. It used three multipliers, three dividers and an adder. The adder adds the outputs of every order.

Finally, the hardware implementation of channel equalizer using RBF neural network structure, show in Fig.4. The chip uses Virtex-2 series 3 million logic gates that Xilinx Company produce. The RBF network structure is two input nodes, eight hidden nodes and an output node.

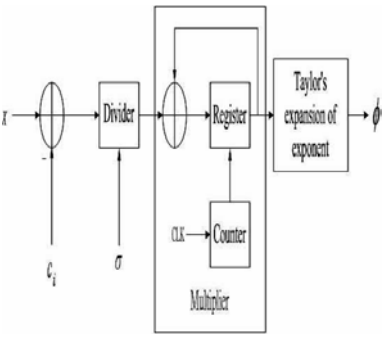

Fig. 2. The structure of gaussian function.

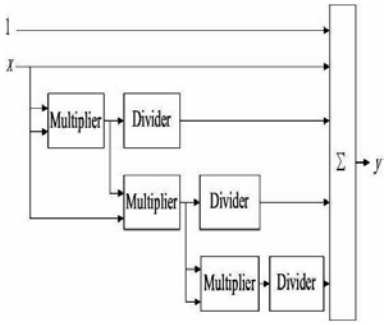

Fig. 3. Taylor expansion of exponential function.

5 Illustrative Examples

A discrete time model of a digital communication system is depicted in Fig.5. A random sequence x_i is passed through a dispersive channel of finite impulse response (FIR), to produce a sequence of outputs $\hat{y}_i$. A term, e_i, which represents additive noise in the system, is then added to each $\hat{y}_i$ to produce an observation sequence y_i. The problem to be considered is that of utilizing the information represented by the observed channel outputs $y_i, y_{i-1}, \ldots, y_{i-m+1}$ to produce an estimate of the input symbol x_{i-d} A device which performs this function is known as an equalizer. The integer's *m* and *d* are known as the order and the delay of the equalizer, respectively. Throughout, the input samples are chosen from $\{-1, 1\}$ with equal probability and assumed to be independent of one another.

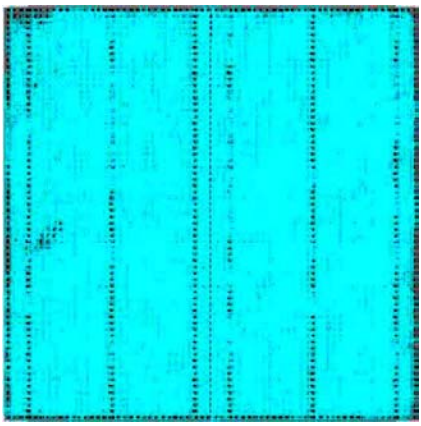

Fig. 4. RBFNN is realized on xc2v3000-5fg676.

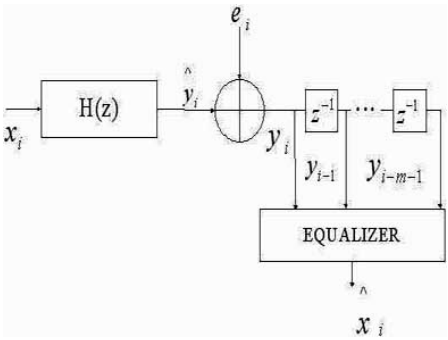

Fig. 5. Schematic of the data transmission system.

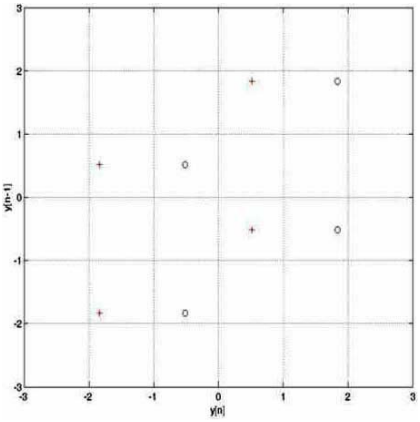

Fig. 6. Desired channel states with no noise.

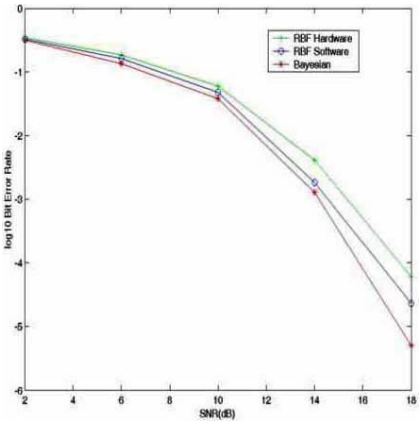

Fig. 7. Comparison of bit error rate curves.

The equalizer performance is described by the probability of misclassification with respect to the signal-to-noise ratio (SNR). With the assumption of independent identically distributed (i.i.d.) sequence the SNR can be defined as

$$SNR = 10log_{10}\frac{\sigma_s^2}{\sigma_e^2} \tag{11}$$

where σ_s^2 represents the signal power and σ_e^2 s the variance of the Gaussian noise.

The equalizer order is chosen as *m*=2. Let the channel transfer function be

$$\hat{y}(n) = (0.5x(n) + x(n-1)) + 0.1 \times (0.5x(n) + x(n-1))^3 \tag{12}$$

All the combinations of *x(n)* and the desired channel states are showed in Fig.6. To see the actual bit error rate (BER), a realization of 10^6 points of sequence *x(n)* and

e(k) are used to test the BER of trained RBF neural network equalizer. It also tests the BER of hardware implementation of the RBF neural network equalizer. The resulting BER curve of the RBF neural network equalizer with software simulation and hardware implementation under the different SNR is show in Fig. 7.

We now compare the performance with the RBF neural network equalizer of software simulation, the RBF neural network equalizer of hardware implementation and Bayesian equalizer. The Bayesian equalizer is near optimal method for communication channel equalizer. Computer simulation results show that the bit error rate of the RBF neural network equalizer is close to the optimal equalizer. But we can see, some BER of hardware implementation is worse. Because the value of the hardware is expressed with binary, so there will be difference in value. If we want to close the BER of original RBF neural network structure, it is necessary to increase the number of bit.

6 Conclusion

The paper describes the RBF neural network structure as channel equalizer. The learning algorithm consists of unsupervised learning and supervised learning, in order to reach a better and a faster training method. And realize its structure of network using FPGA can obtain faster operation efficiency.

In the experiment of time-invariant channel, the result of simulation can be found out, the BER is close to Bayesian equalizer. If we make the BER of hardware simulation closer to original RBF network structure, it can be very easy to increase number of bit.

References

1. Khalid A. Al-Mashouq: The Use of Neural Nets to Combine Equalization with Decoding for Severe Intersymbol Interference Channels. IEEE Transaction on Neural Networks, **5** (1994)
2. S. Chen, Bernard Mulgrew, Peter M. Grant: A Clustering Technique for Digital Communications Channel Equalization Using Radial Basis Function Networks. IEEE Transaction on Neural Networks, **4** (1993)
3. Yutaka, Maeda, Toshiki, Tada: FPGA Implementation of a Pulse Density Neural Network With Learning Ability Using Simultaneous Perturbation. IEEE Transaction on Neural Networks. **14** (2003)
4. Chin Tsu Yen, Wan-de Weng, Yen Tsun Lin: FPGA Realization of a Neural-Network-Based Nonlinear Channel Equalizer. IEEE Transaction on Neural Networks, **51** (2004)
5. J.J. Blake, L.P. Maguire, T.M. McGinnity, B. Roche, L.J. McDaid: The Implementation of Fuzzy Systems, Neural Networks and Fuzzy Neural Networks Using FPGAs. Information Sciences (1998)