in the International scientific conference of Salahaddin University-Erbil (SU-ERBIL2011), which will be held in Erbil, Kurdistan, IRAQ, on October 18-20, 2011.

# FPGA Based Adaptive Neural Equalizer

Thamer M. Jamel
University of Technology
Baghdad – Iraq
thamerjamel@yahoo.com
dr.thamerjamel@uotechnology.edu.iq

Sabah Nasir Hussein
Technical Electrical College
Baghdad – Iraq
drsabah2004@yahoo.com

Ban M. Al-Juboory
University of Baghdad
Baghdad – Iraq

Abstract- This paper presents hardware implementation of an adaptive neural equalizer using Field Programmable Gate Array (FPGA). The adaptive neural equalizer is implemented using a Multi Layer Perceptions (MLP) neural network with the Back-Propagation (BP) algorithm. The network model has a three-layer structure which consists of an input layer, a hidden layer and an output layer. Then the MLP is implemented using FPGA. The simulation results show that the performance of the MLP equalizer using software and hardware implementation were similar, the differences were not much.

The software implementation utilization was 92% of the platform capacity on the FPGA.

Keywords: Neural Network, Adaptive Equalizer, BP MLP, FPGA.

I. INTRODUCTION

Distortion of signals is produced by dispersion in the wireless channel, due to the reception of several paths with different transmission delays. Each of the paths introducing fading in the transmitted signal independently of each other. The multi-path delay spread can commonly be several milliseconds, resulting in simultaneous reception of several separate data bits at the receiver. This distortion is known as inter symbol interference (ISI), and it is the principal cause for the poor performance of a data system over a channel. The adaptive equalizer is used to solve this problem [1].

The reconstruction of binary signals constitutes the major problem in signal reconstruction, arising mainly in digital communications communication system which is shown in Fig. 1. The transmitted sequence \( Y(k) \), which takes binary values \( \{0, 1\} \) with probability that is assumed to be independent of one another, is passed through a dispersive channel. The channel output \( X'(k) \) is corrupted by an additive noise \( N(k) \), which is independent of the transmitted sequence \( Y(k) \). As a result, the observation data \( X(k) \) are equal to \( X'(k) + N(k) \), that is, \( X(k) = X'(k) + N(k) \). The task is to use the information represented by the observed data \( X(k), X(k-1), ..., X(k-m+1) \) to produce an estimate of the values of \( Y(k) \). A device to perform this function is known as an equalizer. Consequently, the reconstruction of binary-value signals is referred to as equalization.

Equalizers are used to overcome the negative effects of the channel [2, 3]. Removing this interference will allow for much more accurate data detection. From the viewpoint of the digital communication and systems, an increasing number of studies can be observed in recent literature with proposed applications of neural networks in a digital communication environment [4,5,6,7,8,9,10,11].

![Figure 1. Schematic of digital communication system](image-url)

These applications largely used neural network because it has superior rejection of noise and better compensation of distortion [12]. Neural networks act as nonlinear maps between received samples and transmitted samples [13].

There are many reasons for implementing a system using a neural network: -

- The main advantage of using a neural network is its ability to adapt and learn.
- Information storage in a neural network simply requires storing the different values of the weights.
- A neural network responds well in the presence of noise.
- A neural network responds well to hardware failure (a change in the value of a certain weight will only affect certain outputs, not all of them) [14].

The key features of neural networks: -

- Asynchronous parallel.
- Distributed processing.
- High-speed computational capability.

The most commonly used tools for the design of signal processing systems are: Application Specific Integrated Circuit (ASIC), Digital Signal Processors (DSP) and FPGA [15]. FPGA has become the first choice for many
digital circuits’ designers [15, 16, 17, 18,19,20,21,22,23,24,25]. In this paper, in order to show the performance of FPGA in digital signal processing applications, an adaptive neural equalizer is implemented on an FPGA.

This paper is organized as follows: section II gives the design of an adaptive neural equalizer based on multi layer perceptions (MLP) with the Back-Propagation (BP) training algorithm. Section III gives the design details of an adaptive neural equalizer using FPGA hardware. Section IV presents the experimental results, in which, the performance of the adaptive neural equalizer in our hardware architecture is shown, and a pure software implementation is introduced and compared with its hardware counterpart.

II. DESIGN OF ADAPTIVE NEURAL EQUALIZER

The neural equalizer based on MLP is built with three layers, called input, hidden and output layer as shown in Fig. 2. There are several units in each layer with \( N_i \) as the number of input neurons, \( N_h \) and \( N_o \) as the number of hidden units and output units. The nodes are connected as shown in Fig.2. The coefficients \( W_{ij}(n) \) and \( W_{jk}(n) \) used indexes \( ij \), while index \( k \) is used for input layer, hidden layer and output layer respectively. Here \( N_i = N_h =3 \) and \( N_o = 1 \) [26]. The equalizer was designed using neural network as shown in Fig. 2.

Each neuron has nonlinear activation function (Sigmoid):

\[
\text{Sigmoid} = \frac{1}{1 + e^{-net}}
\]  

(1)

Therefore, there are twelve weights values that must be calculated using a specific algorithm. The back-propagation algorithm is used in this paper to train the adaptive neural equalizer.

III. DESIGN OF ADAPTIVE NEURAL EQUALIZER USING FIELD PROGRAMMABLE GATE ARRAY TECHNIQUES

This work was performed in three phases, the design phase, simulation phase and implementation phase. Software package (Xilinx Foundation 4.1) has been used to perform the three phases in each design part. Firstly, the schematic design phase, secondly the simulation phase, and thirdly the implementation phase. The hierarchy of the design has three levels as following:-

A. Design of Neural Network

The neuron has two parts that must be considered. The first part consists of a number of multipliers, and adder to perform the following equation:

\[
\text{Net} = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + b
\]  

(2)

Where \( x_1, x_2, \) and \( x_3: \) are the noisy inputs data. \( w_1, w_2, \) and \( w_3: \) are the weights of the neuron, and \( b: \) is the bias of the neuron as shown in Fig. 3 (A).

Three multipliers and three adder circuits are required to perform the above equation. The second part is the activation function of that neuron. In this work, the activation function used for all neurons is the (sigmoid) function. Fig. 3 (B) shows a direct digital hardware implementation of a neuron. The design of the activation function is the second operation part of the neuron. The activation function \( f ( x) \) is used to limit the value of the neuron output to value between 0 and 1. This activation function is given by (1). This equation has been selected because it provides the necessary limiting of the outputs. Unfortunately, this equation contains the transcendental function (exponential), which cannot calculate. Therefore, the approximate equation has been used, which is given below.

\[
F (net) = \frac{1}{2} \left( \frac{net}{1 + \sqrt{\frac{net}{1}}} + 1 \right)
\]  

(3)

Software implementations circuits have been designed to implement the above approximate activation function. These circuits were built using a data flow graph as shown in Fig. 4.
Since the result of this neural network in a digital form, the function of the hardware circuits can be represented as \( f(\text{net}) \) where \( \text{net} \) is the input value.

![A Basic Neuron Model](image)

![A Digital Representation of Neuron](image)

Figure 3. A digital representation of a neuron

The functional units used by this circuit (see Fig. 4) are: two adders, a divider, an absolute value and a divide-by-two circuit. Some simple optimizations have been performed on this circuit. Once the functional unit implemented, it can be suited into FPGA [27].

The curves in Fig. 5 provide a similar limiting function; it is the general characteristics of the sigmoid. This diagram is showing that the two equations (F1 and F2) have the near characteristic.

\[
F_1(x) = \frac{1}{1 + e^{-x}}
\]

\[
F_2(x) = \frac{1}{2} \left( \frac{x}{1 + |x|} + 1 \right)
\]

**B. The Neuron Design**

The complete design of the neuron has been done to create the top of the hierarchy level using the symbol tool. The hierarchy schematic design of the neuron is shown in Fig. 6.

![The sigmoid activation function circuit](image)

![Sigmoid activation functions](image)

![Neuron symbolic representation](image)

Figure 4. The sigmoid activation function circuit

Figure 5. Sigmoid activation functions

Figure 6. Neuron symbolic representation.

Where:
- CE: Is the clock enable to the neuron.
- C: Is the master clock applied to the system.
- CLR: Is the clear pin of the neuron.
- X [7:0]: Is the 8-bit noisy input data.
- XX [7:0]: Is the 8-bit previous noisy input data.
- XXX [7:0]: Is the 8-bit previous of the previous noisy input.
W [8:0]: Is the 9-bit first weight to the neuron.
WW [8:0]: Is the 9-bit second weight to the neuron.
WWW [8:0]: Is the 9-bit third weight to the neuron.
BIAS [15:0]: Is the 16-bit bias of the neuron.
J [7:0]: Is the 8-bit output of the neuron.

The block defines the main inputs and output to the neuron. All numbers that input to the neuron are represented in two’s complements.

C. Design of the Detector Circuit
The detector circuit is needed in the design of an adaptive neural equalizer. The macro detector is shown in Fig. 7:

![Detector Circuit](image)

The detector circuit contains inside it the following circuit that shown in Fig.8:

![Details Circuit](image)

Where the block labeled as (FIVE) is a register contain the value of (0.5). The operation of the detector circuit is to compare the output of the neural network with (0.5). If the output of the neural network greater than or equal to (0.5) then the output of the equalizer equal to one, else the output of the adaptive neural equalizer equal to zero. The complete design of the circuit of the adaptive neural equalizer which contains the neuron and the inputs and output of each neuron and the control circuit for the noisy input signal, weights and bias for each neuron is shown in Fig. 9.

![Complete Design](image)

IV. SIMULATION RESULTS
Fig. 10 shows the beginning operation of the system and how the weights and biases input to the system. Fig. 11 shows the timing diagram for the output of the adaptive neural equalizer when the signal to noise ratio equal to 10dB.

Where:
- CE: is the input signal to enable the overall system.
- CLR: is the input signal to clear the system.
- C: is the clock of the system.

The communication channel used has the impulse response (H) represented in (5)

\[ H(m) = 0.5(1 + \cos \frac{2\pi(m-1)}{w}) \]

Where \( m=0, 1, 2 \) and \( w \) is a scalar factor between 2 and 4. After this time (20\( \mu \)s), the noisy data begins to deliver into the system. The noisy data taken in this example is the same as that used in MATLAB program.

Fig.12 shows the comparison between the output from an equalizer simulated by MATLAB and output from an equalizer designed using FPGA, when the signal to noise ratio 10dB.
in the International scientific conference of Salahaddin University-Erbil (SU-ERBIL2011), which will be held in Erbil, Kurdistan, IRAQ, on October 18-20, 2011.

V. CONCLUSIONS

Adaptive neural equalizer based on MLP structure and learning by BP algorithm was build using FPGA technique. Also MATLAB software package was used for software simulation of adaptive neural equalizer in order to train the neural network, and get the final weights coefficients at convergence state. The hardware architecture is synthesized using the FPGA platform, XC40150xx-9-BG432. The software implementation utilization was 92% of the platform capacity on the FPGA. A comparison between hardware and pure software implementation is then made. Comparison results show that the performance of the MLP equalizer using software and hardware implementation are close to each others.

REFERENCES

in the International scientific conference of Salahaddin University-Erbil (SU-ERBIL2011), which will be held in Erbil, Kurdistan, IRAQ, on October 18-20, 2011.


