Engineering Journal of the University of Qatar, Vol. 15, 2002, pp 135-152

FRAME SYNCHRONIZATION IN OFDM SYSTEMS USING NEURAL NETWORKS, WITH DSP IMPLEMENTATION

Mohamed H. Al-Meer
Electrical Engineering Dept
Engineering College
Qatar University
P. O. Box 2713
Doha – Qatar

Email: almeer@qu.edu.qa

ABSTRACT

This paper presents a method for estimation of the timing offset in OFDM (Orthogonal Frequency Division Multiplexing) systems. The author showed that it is possible to detect the start of the OFDM blocks without the use of pilot tones or the most recently researched the use of correlation with cyclic prefixing, for frame synchronization. The method relies on training the neural networks to recognize a special synchronization pattern transmitted in the beginning of OFDM blocks. The tests and simulation carried out on the system showed an excellent recognition degree in an environment of high AWGN (Additive White Gaussian Noise) noise and multi-path distortion. Finally a real-time DSP (Digital Signal Processing) implementation of the method has been achieved with parallel results to simulation.

1. INTRODUCTION

The OFDM modulation technique has emerged from the last decade with a great reputation. It has been considered as a standard in DAB (digital audio broadcasting) and DTVB (digital TV broadcasting) for wireless applications. OFDM-based discrete multi-tone has been used recently for digital communication on copper networks [1].

The problem in OFDM receiver is the unknown time instant to start the sampling of a new frame. To receive the signal successfully the OFDM receiver must start its FFT at the beginning of the block; otherwise; the orthogonality between sub-carriers is lost. That results the successive received samples to be spread and lost.

Many researchers, who proposed methods to time synchronize OFDM systems; have suggested one of three solutions. First, the use of pilot tones inserted in the data structure, that the receivers can detect, to identify the first sample of the block [2]. Second, transmitting a special synchronization symbol in time domain in order for the receiver to detect it using correlation [3]. Third, an algorithm that relies on the correlation between the data and the cyclic prefix part of the data to extract the time start of the OFDM block [4]. In this paper, the author presents and evaluates a novel method for detecting the start of OFDM block by embedding a synchronization pattern in the start of the block or frame. A neural network in the receiver side detects the presence of the pattern and hence triggers the frame start. The distinct feature of this method is the utilization of the neural network ability to be programmed to function in a hostile environment. Such environment categorized by high degree of attenuation, noise, and multi-path reception.

2. ARTIFICIAL NEURAL NETWORK

It is well known that detection of signals from noisy observations is an important area in statistical signal processing. The associated application extends to Radar, Sonar, and digital communication. In recent years neural networks has been considered for signal detection mainly because it can be trained to operate with acceptable performance, in an environment in which the optimum signal detector is not available [5].

This section describes the most widely type of neural networks for recognition purposes, the feed forward neural network. The general type of feed-forward multi-layer perceptron consists of three layers, input layer, hidden layer, and output layer. All of these layers are structured from a processing element called the neuron or the perceptron. The neurons in each layer are fully interconnected with others in the neighborhood layers as shown in figure 1. A neuron receives a number of inputs x_1, x_2, ... , x_n which are multiplied by a set of weights, w_1, w_2, ..., w_n, and the results are summed. A constant threshold is also added to the sum. Then the output signal is generated according to some predefined activation function, in our ANN model, the logistic function is used.

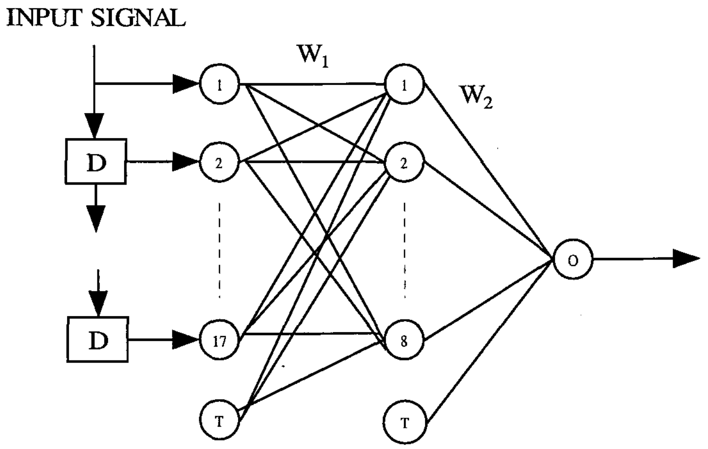

Fig.1 Multi layer perceptron for the frame detector model

The outputs of the neurons in the input layer are computed by presenting an input vector to the input layer and using the sum of products with the weight associated and activation functions. The output of the neurons in the input layer then becomes inputs to the next layer, and so on. The output of the network is therefore the output of the neurons lying in the output layer. Figure 1 shows the three-layer feed forward neural network with delay elements inputs feeding the main network inputs.

To serve the purpose of frame synchronization, a number of tests were carried out by neural network simulators in order to determine the best structure and the optimal number of neurons for both hidden and output layers. All of the hidden and output layer neurons that permit to take into account all the performance required by frame synchronizer, is chosen during training phase.

According to the tests, author have chosen for the synchronizer, the number of layers to be three, the input layer to consist of 17 neurons (coincide with Synchronization pattern elements), and the hidden layer neurons to be 8. Finally the output layer has been chosen to have only one neuron to represent the decision state of 0 or 1. The activation function for every neuron is selected to be sigmoidal activation function with a boundary limit of [0, 1].

In order to feed the network with 17 images of the inputs signal, it has been decided to base the structure of the model on the time delay. In this structure N-1 delay elements are cascaded one after the other. In effect, there is only one input to the network, which represents the current sample emerging from analog to digital converter. Starting from this input, other N-1 delay version of the input are generated, by taking the outputs of the cascaded delay elements and feed them to the neuron inputs.

3. SYNCHRONIZATION SIGNAL

The synchronization pattern is constructed from a complex function known as Wobbulation signal [6]. The signal is defined as

$$x_i = e^{j\pi i^2 / N} \tag{1}$$

Where i=, 2… N-1, we have chosen N=16 but the sequence is extended from 0 to 16 giving an evenly symmetrical function with 17 values. Only the real part of the generated sequence is considered:

This synchronization symbol is categorized with:

1. This synchronization symbol has a strongly distinguishable criterion in order to be unlikely to occur by chance in the data symbols.

2. It own strong and sharp autocorrelation function to minimize the autocorrelation side loops; in order to minimize a false synchronization probability with noise.

4. TRAINING

The central issue in the application of every recognition technique is the selection of the most appropriate signal representation. As a result, the structure of the input pattern used in training and testing the neural network system, have to be chosen properly.

The most appropriate signal presentation for training is based on FIFO buffer structure. 17 locations of a type FIFO, the first data enters the buffer will be the first one leaves it, is considered. Now the purpose of that buffer structure is to detect the synchronization pattern as it passes through the buffer. If we imagine the synchronization pattern inside the buffer completely aligned, then this should represent for us a case of true recognition where it will be give a numerical value of 0.9. Similarly, if we have any slight misalignment of the pattern relative to the buffer, then this should represent to us no recognition case where it will be given a value of 0.1 to its output signal. As a result, the different alignment images of the synchronization pattern with their given values representing the recognition states will be subjected for training. This eventually will help in distinguishing between the correct and the incorrect synchronization patterns.

The author has recommended three different buffer-synchronization signal representations to train the network with, the no-shift, the half-shift, and finally the complete-shift. Those can enhance the recognition process as a result.

No-Shift

This configuration assumes that the input is stationary; that is, only one image (a snapshot) of the synchronization pattern is considered, although the pattern samples enter the buffer in sequential order, which creates many shifted images of the main pattern. Therefore, the network is trained to identify the complete image of the synchronization pattern as a train vector. The detector, when presented with the signal completely aligned in its buffer, which has exactly the same length as the synchronization signal, it should trigger a "correct recognition" and consequently gives it an output value of "0.9". This configuration has been named "no shift" configuration.

Half-shift

The second configuration assumes training of the network using the Synchronization signal as it enters the buffer in sequence. Whenever the synchronization signal advances one sample after another, each image or snapshot is considered one of the training vectors and it will be assigned "negative recognition" and will be given an output value of "0.1". The process continues until the Synchronization signal aligns exactly with the buffer without any advanced or delayed shifts, then its output will be assigned "correct recognition" and hence will be given the output value of "0.9". The rest of the signal images, as it leaves the buffer will not be counted in order to reduce the training overhead. This type of input configuration will be named "half shift" configuration.

Complete-shift

The third configuration is obvious, as it resembles the previous one except the image of the Synchronization signal while it leaves the buffer after a correct alignment will also be taken into account; it will be assigned "negative recognition" and will be given an output value of "0.1". As a result, this configuration is being named "complete shift" configuration.

Although these three configurations are simple to implement in constructing the training vector, they increase the capability of the network to distinguish between correct synchronization alignment and false ones.

After the training vectors have been constructed, the neural network is then trained according to the different input configurations. The resulted weights and biases are then saved and stored to be used in the next phase.

The next phase, the testing phase, then starts. Here the network is presented with sequences of synchronization pattern in order to assess its behavior. 1000 synchronization pattern are constructed in sequential form and contaminated deliberately with the three deformation caused by the wireless channel, AWGN, multi-path, and attenuation. These sequences are then presented to the network under test to assess its function.

The last phase in the design process is the utilization of the target weights and biases in a DSP platform. Consequently, the operation of the neural network in real time is accomplished. The flowchart depicted in figure 2, shows different phases of the process of synchronizer design.

5. DSP REALIZATION OF THE SYNCHRONIZATION ALGORITHM

The DSP used for the model realization is the TMS320C50 of 50MHz and 25 MIPS (Million instructions per second). The processor constitutes the central part of the

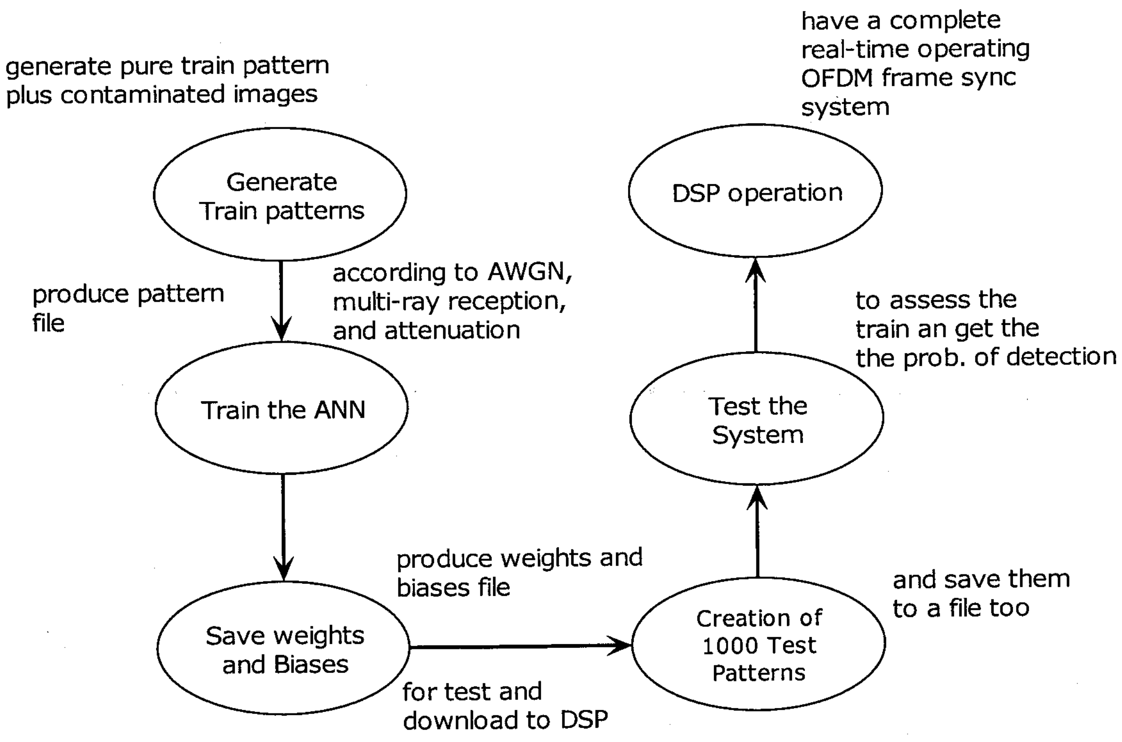
generate pure train pattern plus contaminated images
Generate Train patterns
produce pattern file
according to AWGN, multi-ray reception, and attenuation
Train the ANN
Save weights and Biases
produce weights and biases file
for test and download to DSP
Creation of 1000 Test Patterns
and save them to a file too
Test the System
to assess the train an get the the prob. of detection
DSP operation
have a complete real-time operating OFDM frame sync system

Fig 2. The different phases for frame synchronizer design

teaching kit DSK-C50. Apart from the processor, the teaching kit also comprises of analog – to – digital converter, digital – to –analog converter and serial communication hardware to connect with personal computers. The DSP with its one clock pulse instruction execution and MAC (Multiply Accumulate) instruction is considered an efficient microcomputer machine to solve math-intensive algorithms.

DSP Neural Network weights realization

The synapse weights wij are stored in a one-dimensional buffer. The use of the array instead of a matrix makes the processing more efficient because it needs one access instead of two required from a matrix. Figure 3 shows the order sequence of the stored synapse weights.

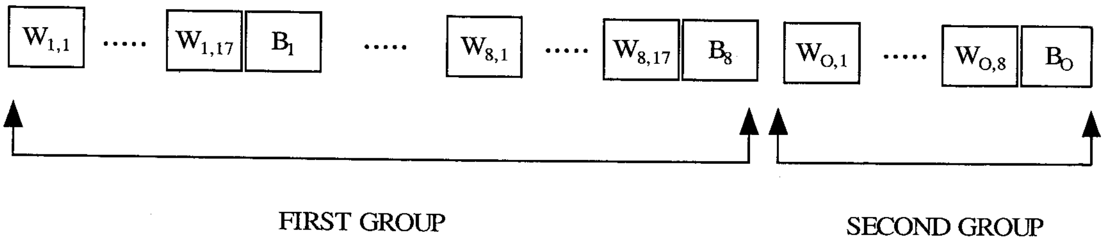

Fig.3 Sequence of synapse weights and biases stored in one dimensional buffer

All the weights are divided into two groups according to the ANN structure. The first groups consist of the synapse weights connecting the input neurons with the hidden neurons, and the second group consists of the weights connecting the input neurons with the output neurons. The first group is subdivided to 8 sub-groups, each one made up of 17 synapse weights connecting each of the 17 input neurons with the 8 hidden layer neurons. The total number of synapse weights in the first group is equal to 136 plus 8 threshold weights for each of the hidden layer neurons.

The second group is organized into one division comprising 8 synapse weights connecting the hidden layer with the only output layer the network owns. Therefore the total number of synapse weights of the second group is equal to 8 plus one threshold weight for the output neuron. In this manner, a saving of memory occupancy up to 50% has been achieved.

The program in the receiver which executes the ANN calculations accepts the input from its buffer and performs a series of multiplication by the weights of the first group producing 8 results which will be saved later in a special array in memory. The equation for the hidden layer neurons and the output neuron will be:

$$O(i) = \sum_{j=1}^{M} wij \bullet x(j) + B_i,,,,,,,,,i = 1,...,L \qquad (2)$$

Where B_i is the bias weight for every neuron used, and $x(i)$ is the input to the current layer. The $O(i)$ is then substituted in the activation function which only could be realized using a look-up table method. This method is discussed in the following section.

DSP sigmoidal activation function generation

The mathematical realization of the neural network structure comprises a series of multiplication and addition operations which are simple to implement using a DSP processor designed for such computational loads. But the main obstacle to this implementation seemed to be the activation function found in each neuron. DSP processors are not equipped with specialized units for computing such a function; so, alternative methods should be followed to fulfill this demand. The alternative used in this implementation will be the look-up table method.

The idea of a look-up table is simple. A general-purpose computer is utilized to select a series of numbers with predefined range that should be substituted in the activation function

$$f(x) = \frac{1}{1+e^{-x}} \qquad (3)$$

which generates the required output. This range of outputs is downloaded to the DSP memory that implements the neural network.

DSP-based Gaussian Noise Generation

An inexpensive, 16-bit, fixed point DSP chip such as the one used here can generate real-time Gaussian noise signal for testing the performance of frame synchronization technique in the presence of noise [7]. The linear congruential generator was used to generate first a uniformly distributed random number sequence given as follows:

$$X(n+1) = \left(a.X(n)+c\right) \;\; Modulus \;\; m \qquad (4)$$

Where a, c, and m are constants; modulus is the modulus operator; n>=0; and X(0) is the seed or the starting value. And for properly chosen constants and a given seed value, this algorithm produces a uniformly distributed random sequence of positive integers between 0 and m. If m is chosen to be the word size of the DSP chip used which is 16, then the modulo m operation becomes quite simple. The operation X modulus m is accomplished by taking m LSB (least significant bits) of

One method of generating a Gaussian distributed random number sequence is to apply the central limit theorem to batches of uniformly distributed random numbers. This idea is expressed mathematically as:

$$g(n) = \frac{1}{B}\sum_{i=0}^{B-1} x(i) \qquad (5)$$

Where x(i) are uniformly distributed, independent random numbers; B is the batch size (16) and g(n) is the approximated Gaussian sequence. This algorithm produces a random sequence whose distribution is close to Gaussian, provided that the batch size B is greater than 10.

To generate the random noise, some important points have to be considered:

1. The batch length was chosen as 16, and every time the uniform random number is generated, it will be divided by 256 which is 8 times shift to the right. This case was considered to avoid overflow while batching.

2. Adding the value to itself a number of times selected by the user enlarge the generated Gaussian random number. This step was carried out in order to generated noise with different amplitude and hence with different RMS values. This RMS value is an important figure in calculating the SNR when Synchronization symbol transmitted contaminated with such noise.

DSP-based Multi-path effect generation

The addition of the second path propagation to the main path is an important simulation criterion for training and testing the neural network. The objective here is to withstand reception of Synchronization symbols in channels where such deformation exists. Nevertheless, it is also important to implement such simulation techniques in a real-time DSP platform to justify its practical use. That is why an algorithmic solution for implementation is considered next.

The developed DSP code is based on creating an array of 20 memory locations. This array is capable to shift all of its content one location up and it is also able to save the most recent location with the sample coming from the analog to digital converter.

The procedure starts by invoking a new sample from analog to digital converter then it is pushed in to the buffer. The previous images of the samples are also accessible up to 20 previous samples per a time. Any one of these past samples can be taken by halving its amplitude then adding it to the current sample.

6. RESULTS AND DISCUSSION

The training has been done for the recognition of the synchronization pattern with the inheritance of additive white Guassian noise AWGN. The typical SNR (Signal to Noise Ratio) of the training pattern was 8.6 dB; although other SNR values for training were 5.6 dB, 3.4 dB, and 1.9 dB. Accordingly, the test SNR range was 20 dB to 2dB. Two jüdgment criteria were considered, the probability of detection and the probability of false alarm

The first question arises that this part of the result answers is " Is the neural network frame detector capable to be trained to work in a noisy system, and if the network is trained to work in more noise level, does it help to have a better recognition performance or not?".

To answer this question, a graphical comparison is created. This graphical comparison is based on a ANN frame detector model where 3 different noise levels contaminating the main Synchronization test symbol in a no-shift training sequence. In addition, 500 positive recognition patterns are used in the training phase to represent the "signal presence" mode and another 500 negative recognition are used to represent "signal absence" mode. The 3 SNR the test symbol trained with are, 5.6 dB, 3.4 dB, and 1.9 dB. The results are shown in Figure 4.

From this graphical comparison it can be seen that "a neural network designed to recognize a test symbol for frame synchronization purpose is quite capable with the help of training in a contaminated Guassian noise to have correct recognition than training in non-Guassian noise environment. In addition, whenever the noise level is increased, a better performance is reached".

Now the training sequence strategies will be discussed. As it is known previously, there are three sequences in which the test symbol is identified, no-shift, half-shift, and complete-shift. The next graphical presentation shows the detection response under noise contamination for the three shapes of sequences in order to see which the best is.

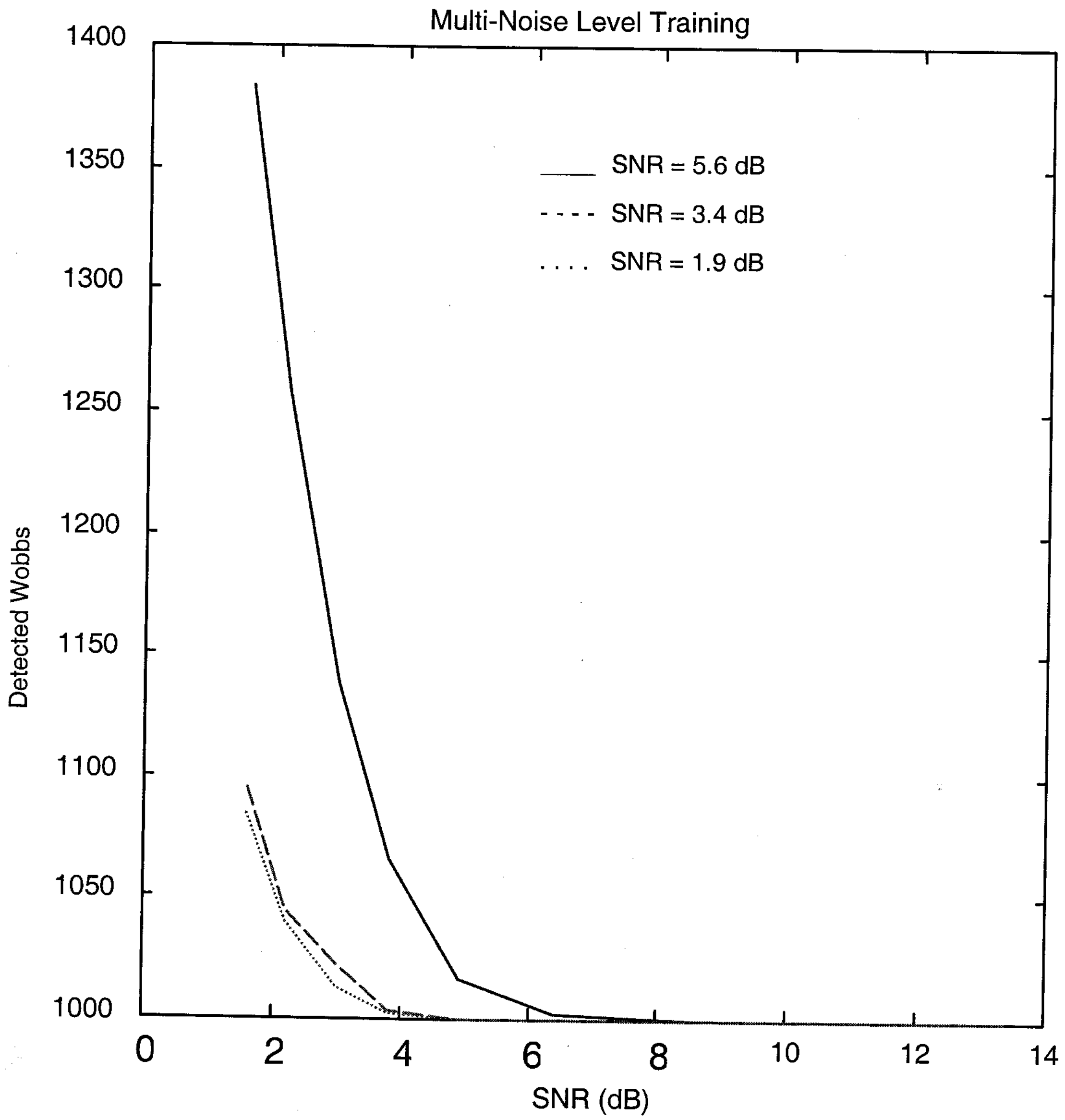

Fig. 4 System trained under different noise levels

All of the curves shown in Figure 5 represent a system trained under a Guassian noise of SNR 5.6 dB. The graph shows a solid line, dashed line, and a dotted line for no-shift, half-shift, and complete-shift sequences respectively.

The complete shift gives the best response of the detector even though the half-shift sequence is better than the no-shift sequence.

Now the multi-path effect will be discussed and by analyzing the next graphical presentation, Figure 6, where it shows a plot of the non-trained and the trained network for such a second ray reception. The dashed line denotes the untrained network performance whereas the dotted line represents the trained network. Moreover, the second path reception is chosen to be of delay 8 samples and of a gain of 0.5. Now the multi-path effect may degrade reception seriously, but by training the frame synchronizer based on the neural network to detect its target in that type of hostility, a better detection performance is achieved.

In addition to the last three results discussed The DSP implementation of the frame start detection for OFDM modem synchronization is successfully accomplished in this thesis and accordingly test results are generated as shown in earlier section, moreover to know more detailed performance assessment, the next graphical presentation is advised.

Figure 7 shows a plot for the half-shift sequence network for both the simulated and the real-time DSP-implemented cases. The training was carried out for both cases in no-noise environment. Figure 8 shows the performance plot for both the simulated and the DSP-implemented cases when complete-shift sequence training is adopted, in addition the training is noise-free.

The DSP implementation results showed satisfactory similarity with the simulation results for the frame detector model performance. Therefore it can be stated that, "the real-time implementation and the simulation results proved the possibility of neural network application in frame synchronization for OFDM systems".

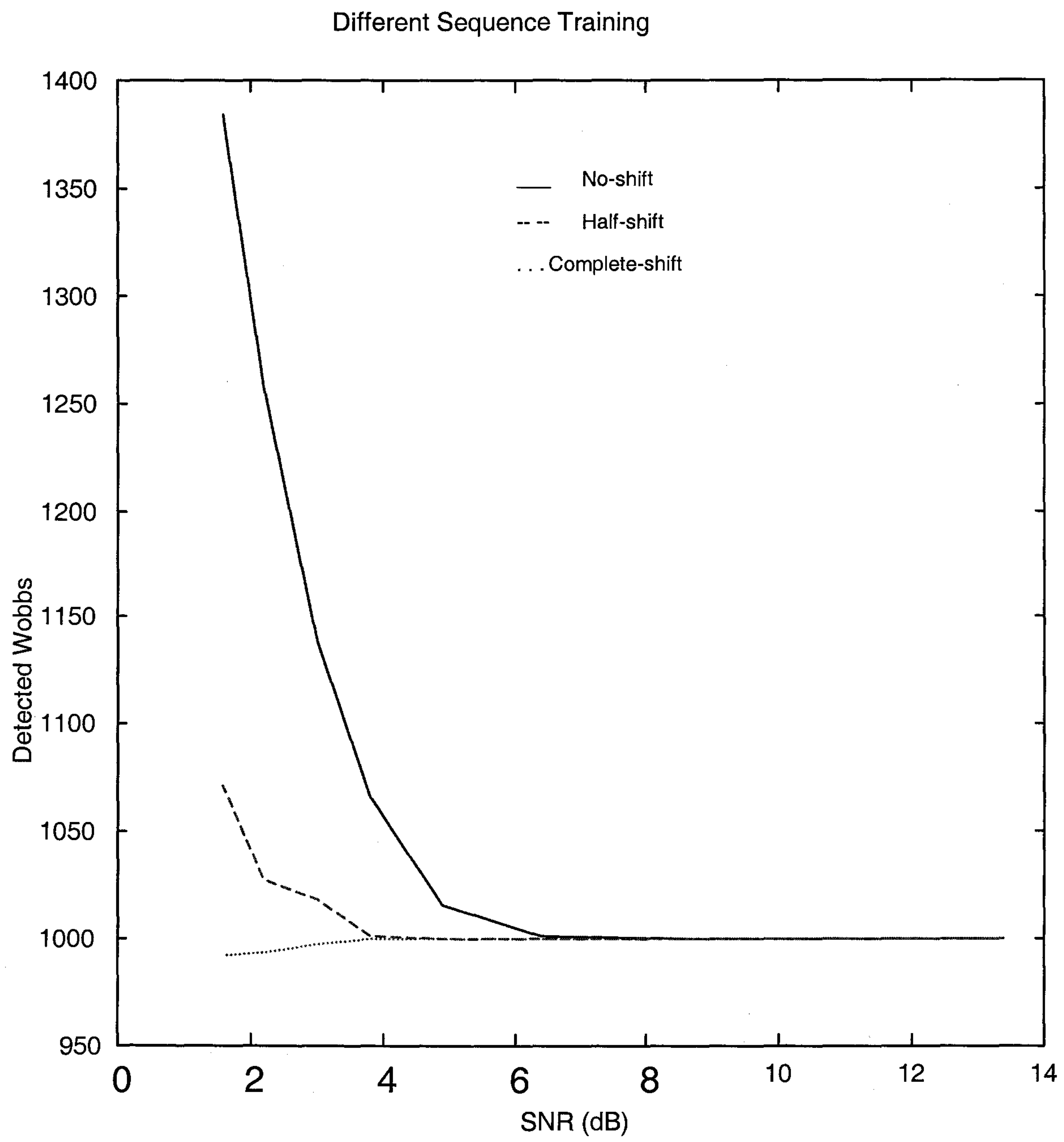

Fig. 5 System trained under different recognition sequences

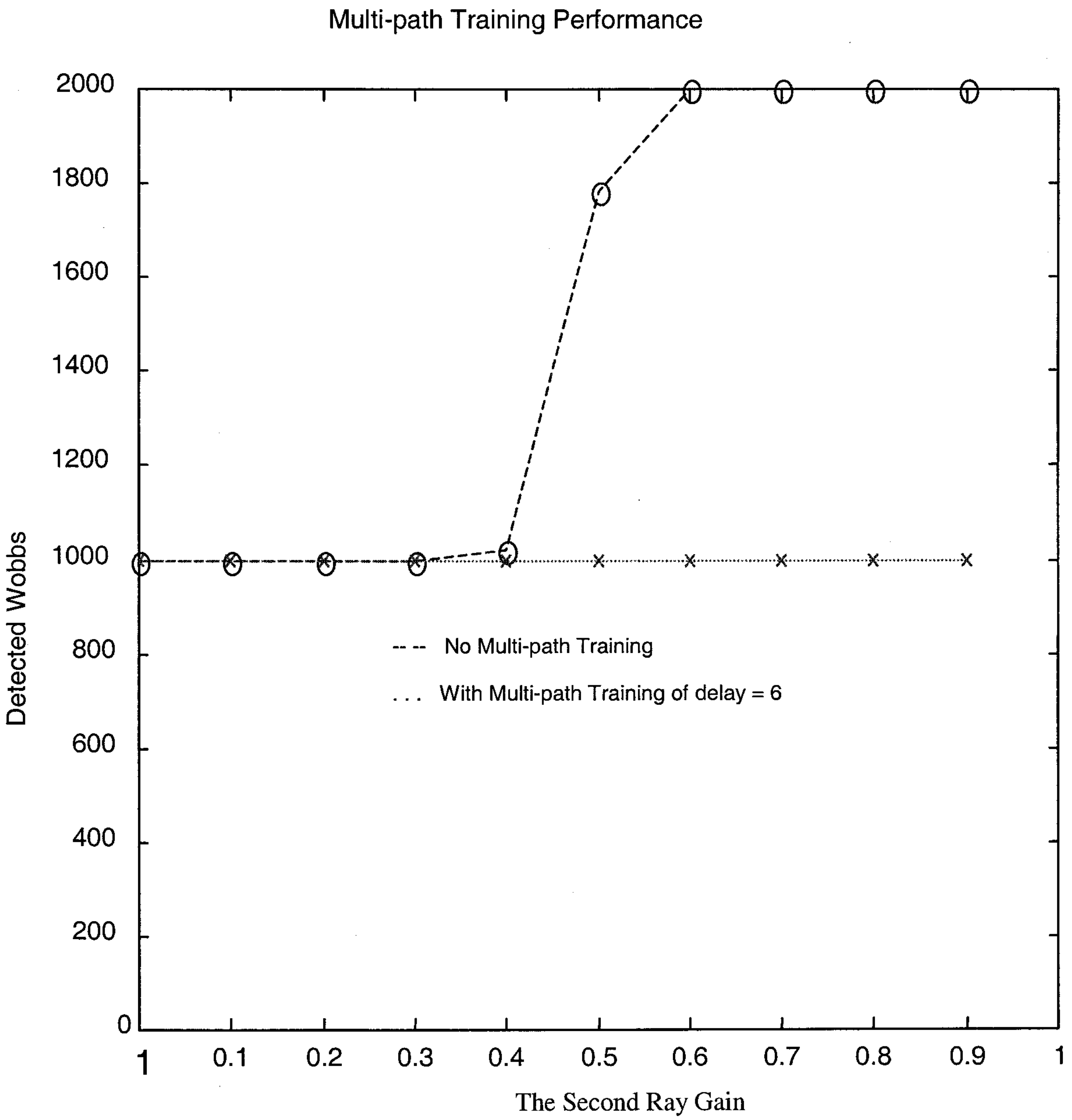

Fig.6 System trained with and without multi-path recognition

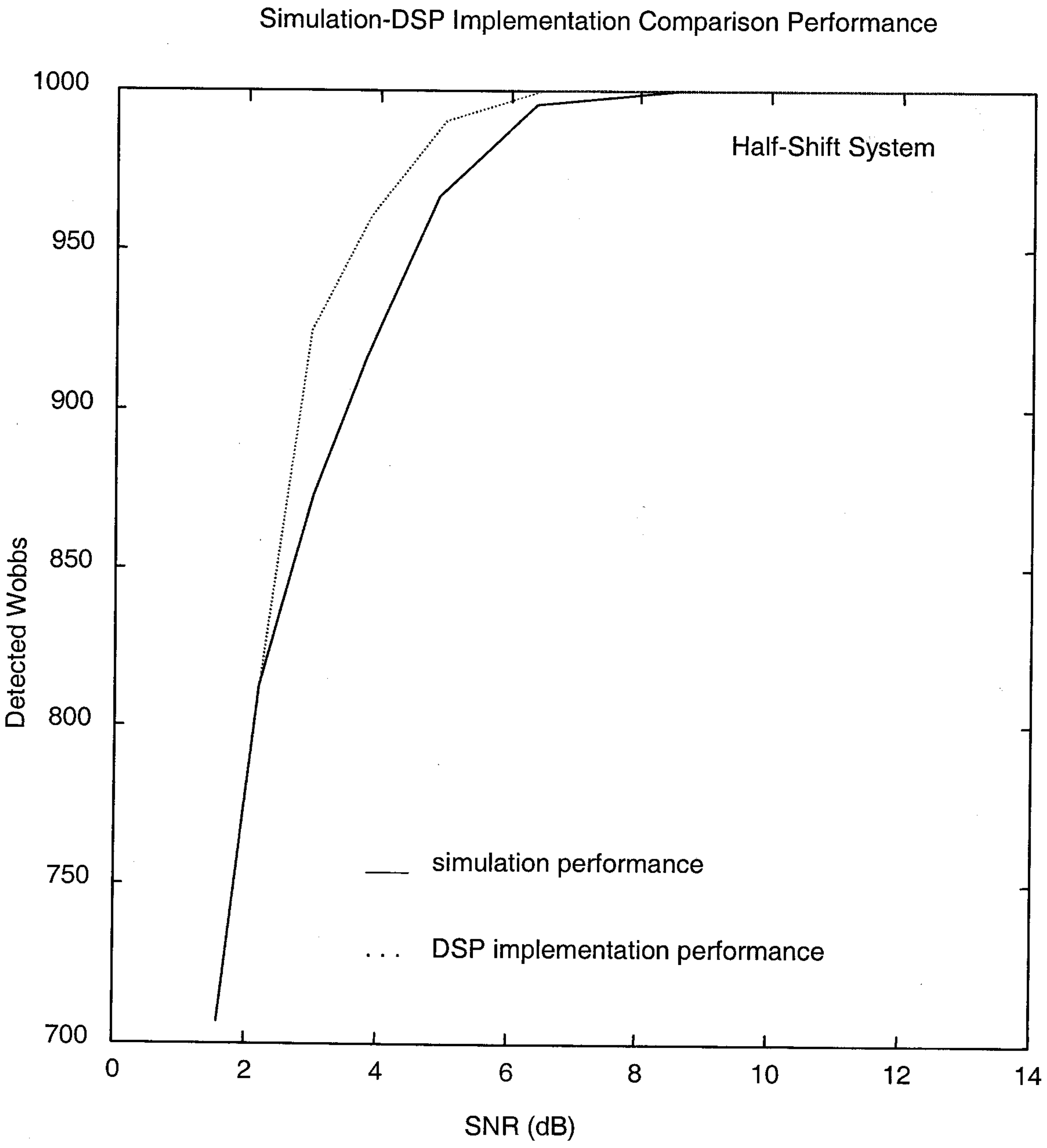

Fig.7 Comparison between performances of simulated and real time DSP implementation half-shifted systems with threshold = 0.75

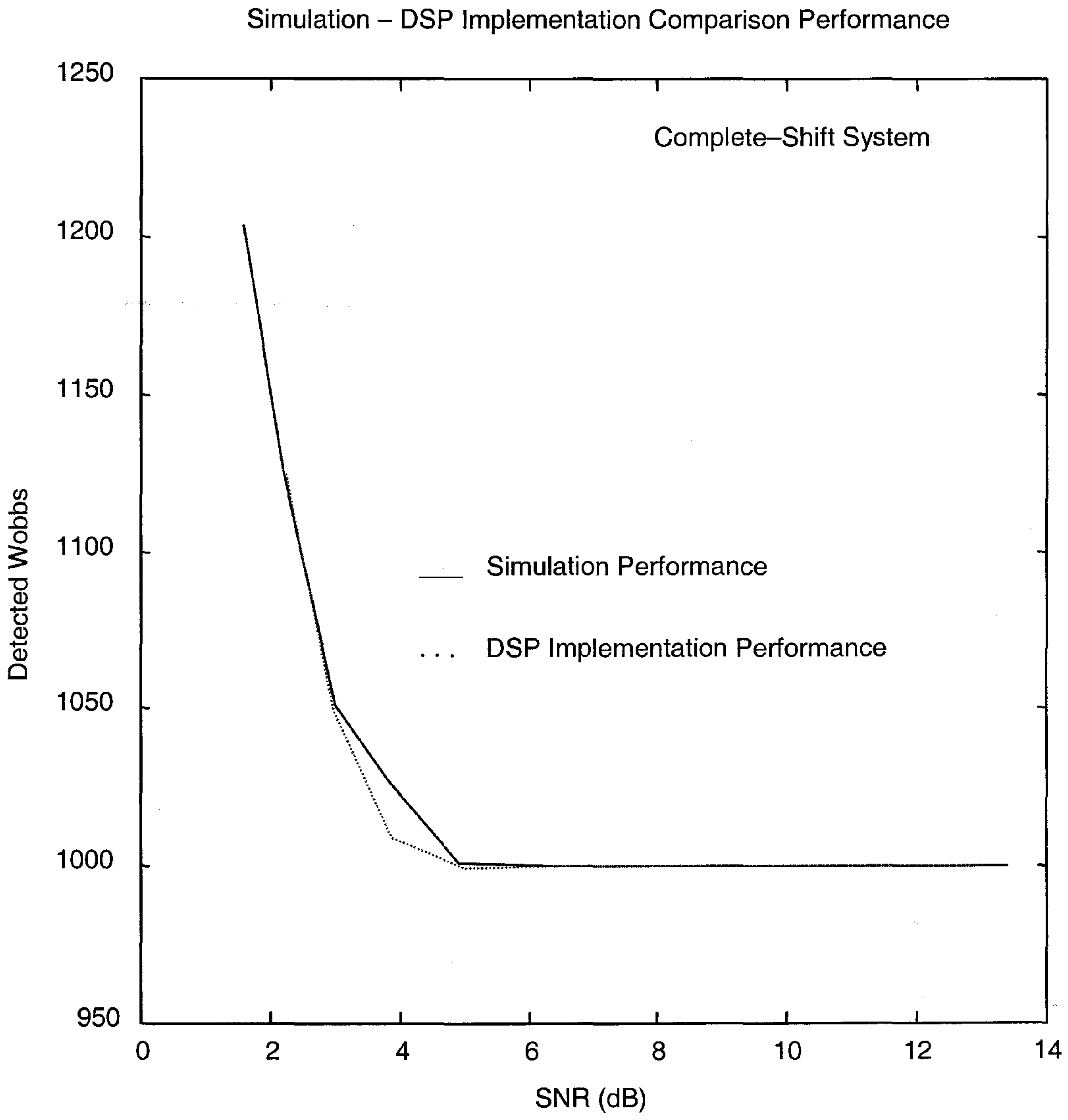

Fig. 8 Comparison between performances of simulated and real time DSP implementation complete-shift systems with threshold = 0.5

7. CONCLUSION

In this paper we have presented a frame synchronizer or detector in OFDM systems which do not need pilots but uses the principle of neural networks to identify the synchronization pattern at the receiver. It is derived under the assumption that the channel is AWGN and time-dispersive (multi-path effect). These two criteria are the basis for training the network to operate with. Simulations showed that the frame synchronizer, after training, has succeeded to identify the synchronization pattern in a SNR reaching 2 dB.

REFERENCES

1. **J. F. Helard, January 1999**. Time synchronization without specific symbols for OFDM, Electronics Letters, Vol. 35, No. 2, pp. 130-131.

2. **W. D. Warner, August 1993.** OFDM/FM frame synchronization for mobile radio data communications, IEEE Transaction on Vehicular Technology. Vol. 42, No. 3, pp. 302-313.

3. **T. M. Schmidl, December 1997.** Robust frequency and timing synchronization for OFDM, IEEE Transaction on Communication, Vol. 45, No. 2, pp. 1613-1621.

4. **D. Landstorm and VanDeBeek j. j., October 1997.** Time and frequency offset estimation in OFDM systems employing pulse shaping, Proceedings of IEEE International Conference on Universal Personal Communication (ICUPC 97), pp. 279-283, San Deigo, California, USA.

5. **Z. Michaloulou, March 1995**. Performance evaluation of multilayer perceptron in signal detection and classification, IEEE Transaction on Neural Networks, Vol. 6, No. 2, pp. 381-386.

6. **A. Eyadeh, May 1997.** Synchronization techniques for an OFDM-based indoor wireless data communication systems, Ph.D. thesis, university of Wales Swansea, Electrical engineering dept.

7. **B. Salibrici, April 1993.** Fixed point DSP chip can generate real time random noise, EDN magazine, Vol. 38, No. 9, pp. 119-122.