Adaptive Filter Application in Echo Cancellation System and Implementation using FPGA

Rafid Ahmed Khalil

Electrical Engineering Department, Engineering College, University of Mosul
E-mail: rafidamori@yahoo.com

Abstract

In telephony system, the received signal by the loudspeaker, is reverberated through the environment and picked up by the microphone. It is called an echo signal. Which is in the form of time delayed and attenuated image of original speech signal, and causes a reduction in the quality of the communication. Adaptive filters are a class of filters that iteratively alter their parameters in order to minimize a difference between a desired output and their output. In the case of acoustic echo, the optimal output is an echoed signal that accurately emulates the unwanted echo signal. This is then used to negate the echo in the return signal. The better the adaptive filter simulates this echo, the more successful the cancellation will be. This paper examines LMS algorithm of adaptive filtering and the application in acoustic echo cancellation system. Employing a discrete signal processing in Matlab for simulation with real acoustic signals. Also a hardware implementation of an adaptive filter have been developed using XC3S500E Xilinx FPGA chip, and VHDL language on RTL abstraction level.

Keywords: Acoustic echo cancellation, Adaptive Filter, FPGA, VHDL.

[illegible]

[illegible]

[illegible]

[illegible] LMS [illegible] Matlab [illegible] XC3S500E Xilinx [illegible] FPGA [illegible] VHDL [illegible] RTL [illegible]

Received 31 Oct. 2007 **Accepted 16 Dec. 2007**

1. Introduction

Digital Signal Processor (DSP) and Application Specific Integrated Circuits (ASICs) have traditionally been the common means for building and implementing Adaptive Filters. However such computing paradigms suffer from the constant need of establishing a trade-off between flexibility and performance. Due to the technological advance in the development of programmable logic devices, Field Programmable Gate Array (FPGA) have become attractive for realizing adaptive filters. FPGAs exhibit excellent flexibility in terms of reprogramming the same hardware and at the same time achieving good performance by enabling parallel computation at short processing time.

Acoustic echo occurs when an audio signal is reverberated in a real environment, resulting in the original intended signal plus attenuated, time delayed images of this signal. The echo cancellation scheme is depicted in Figure (1). Here the echo path is the 'plant' or 'channel' to be identified. The goal is to subtract a synthesized version of the echo from another signal (for example, picked up by a microphone) so that the resulting signal is 'free of echo' and really contains only the signal of interest. A simple example is given in Figure (2) to clarify things [1, 2].

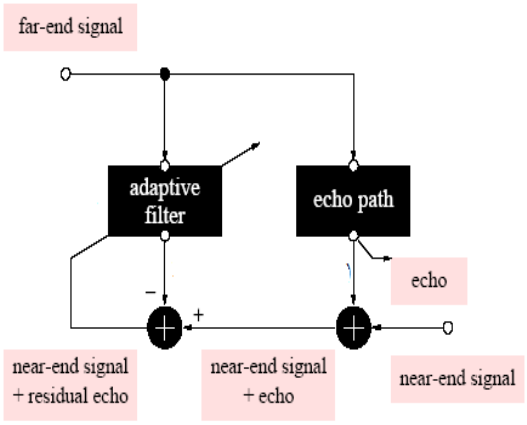

Figure (1) Echo cancellation scheme.

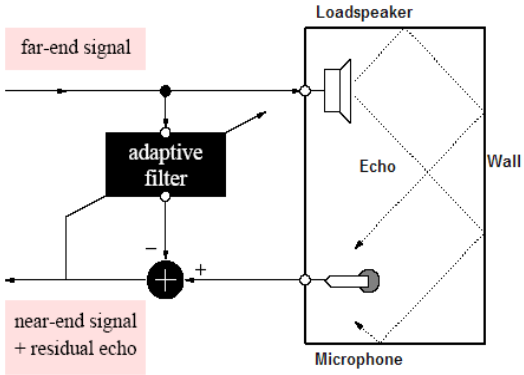

Figure (2) Acoustic echo cancellation

This scheme applies to hands-free telephony inside a car, or teleconferencing in a conference room. The far end signal is fed into a loudspeaker (mounted on the dashboard, say, in the hands-free telephony application). The microphone picks up the near-end talker signal as well as an echoed version of the loudspeaker output, filtered by the room acoustics. The desired signal (see Figure (1) again) thus consists of the echo ('plant output') as well as the near-end talker signal. It is assumed that the near-end signal is statistically independent of the far-end signal, which results in the adaptive filter trying to model the echo path as if there were no near-end signal. The filter weights are adjusted principally in those periods, hen only the far end party is talking. In these periods, the error signal is truly a residual echo signal, and hence may indeed be fed back to adjust the filter. Recall that the adaptive filter has an adaptation and a filtering process. The filtering process is run continuously, even in the presence of the near-end talker, to remove the echo. It only the adaptation of the filter weights that gets switched off. Such a scheme clearly requires an extra circuit that can detect when the near-end talker is speaking [1-3].

The method used to cancel the echo signal is known as adaptive filtering. Adaptive filters are dynamic filters which iteratively alter their characteristics in order to achieve an optimal desired output. An adaptive filter algorithmically alters its parameters in order to minimize a function of the difference between the desired output $d(n)$ and its actual output

$y(n)$. This function is known as the objective function of the adaptive algorithm. Figure (3) shows a block diagram of the adaptive echo cancellation model. Where the $x(n)$ is input signal, the filter $H(n)$ represents the impulse response of the acoustic environment, $W(n)$ represents the adaptive filter used to cancel the echo signal. The adaptive filter aims to equate its output $y(n)$ to the desired output $d(n)$ (the signal reverberated within the acoustic environment). The external noise input $no(n)$ is neglected here. At each iteration the error signal, $e(n) = d(n) - y(n)$, is fed back into the filter, where the filter characteristics are altered accordingly [1-4].

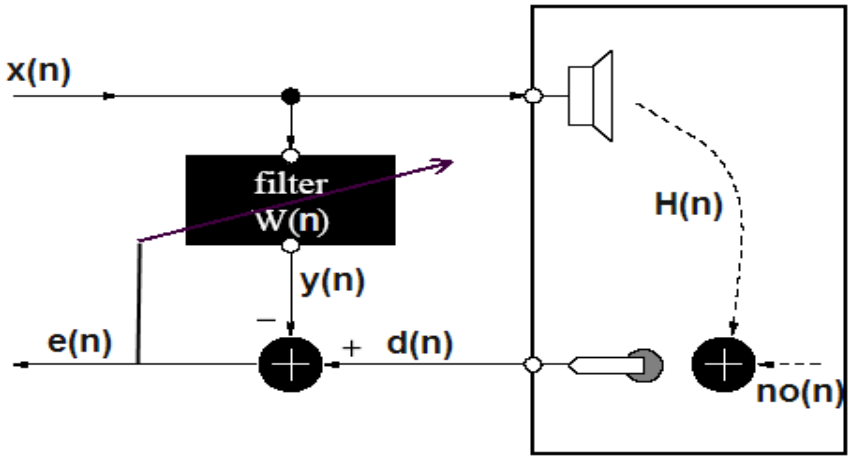

Figure (3) Echo cancellation set-up

As literatures review: C. Choo, et. al. [5], discussed hardware implementation of NLSM adaptive filtering system on FPGA for imbedded systems. R. Dony and his colleague [6], presented FPGA Implementation of the LMS Adaptive Filter for Audio Processing. The research in [7], presented A comparison study between the costs (silicon area) with regards to the, speed and required computational resources for different adaptive algorithms. The goals of this work is to examine adaptive filtering LMS technique as it apply to acoustic echo cancellation. Simulate this adaptive filtering algorithm and acoustic echo cancellation system using Matlab with real acoustic signal. Finally, performing hardware implementation of adaptive filter using the Spartan-3E XC3S500E Xilinx FPGA chip, and VHDL hardware description language with Xilinx ISE 8.2i Software.

The rest of this paper is divided into the following sections. Section 2 deals with acoustic signal processing theory in adaptive filters. Section 3 presents the basis of adaptive filtering technique as well as the development and derivation. Section 4 details the simulations of adaptive filtering technique and acoustic echo cancellation system as developed in Matlab. This section shows the results of these simulations as well as discussing the advantages and disadvantages of this technique. Section 5 outlines the hardware implementation of an adaptive filter using the XC3S500E Xilinx FPGA chip on Spartan-3E starter development kit, the details of which are also examined in this section. Section 6 gives the conclusions.

2. Acoustic Signal processing

2.1 Random Signals

The input vector of the acoustic echo cancellation system are unknown before they arrive. Also it is difficult to predict these values, they appear to behave randomly. A random signal, expressed by random variable function, $x(t)$, does not have a precise description of its waveform. It may, however, be possible to express these random processes by statistical models. A single occurrence of a random variable appears to behave unpredictably. But if we

take several occurrences of the variable, each denoted by n, then the random signal is expressed by two variables, $x(t,n)$.

The main characteristic of a random signal treated, is known as the expectation of a random signal. It is defined as the mean value across all n occurrences of that random variable, denoted by $E[x(t)]$, where $x(t)$ is the input random variable. It should be noted that the number of input occurrences into the acoustic echo cancellation system is always 1. Throughout this work the expectation of an input signal is equal to the actual value of that signal. However, the $E[x(n)]$ notation shall still be used in order to derive the algorithm used in adaptive filter [2,9].

2.2 Stationary Signals

A signal can be considered stationary, if the two following criteria are met [2,8]:

1. The mean values, or expectations, of the signal are constant for any shift in time.

$$m_x(n) = m_x(n+K)$$

2. The autocorrelation function is also constant over an arbitrary time shift.

$$\phi_{xx}(n,m) = \phi_{xx}(n+k,m+k)$$

The above implies that the statistical properties of a stationary signal are constant over time. In the derivation of adaptive filtering algorithm it is often assumed that the signals input to the algorithm are stationary. Speech signals are not stationary in the wide sense, however it exhibit some temporary stationary behavior, as it will be seen in the next section.

2.3 Speech Signals

A speech signal consists of three classes of sounds. They are voiced, fricative and plosive sounds. Voiced sounds are caused by excitation of the vocal tract with quasi-periodic pulses of airflow. Fricative sounds are formed by constricting the vocal tract and passing air through it, causing turbulence that results in a noise-like sound. Plosive sounds are created by closing up the vocal tract, building up air behind it then suddenly releasing it, this is heard in the sound made by the letter p [2]. Fig. (4), shows a time representation of a speech signal. That is, its mean values vary with time and cannot be predicted using the above mathematical models for random processes. However, a speech signal can be considered as a linear composite of the above three classes of sound, each of these sounds are stationary and remain fairly constant over intervals of the order of 40 ms [2,10].

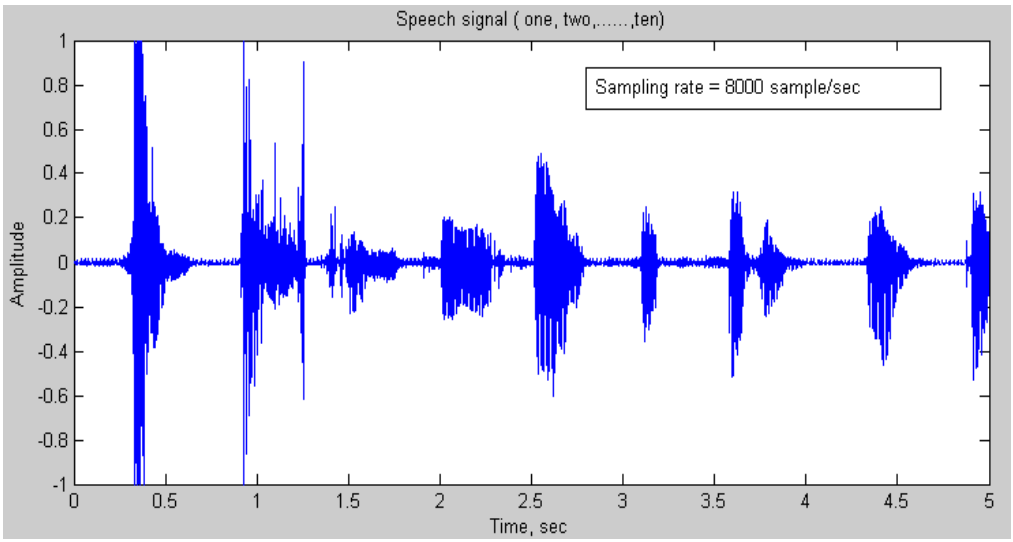

Figure (4) Speech signal representation.

The theory behind the derivations of many adaptive filtering algorithms usually requires the input signal to be stationary. Although speech is non-stationary for all time, it is an assumption of this research that the short term stationary behavior outlined above will prove adequate for the adaptive filters to function as desired [12,13] .

3. Adaptive Filters

3.1 Structure of Adaptive Filter

Figure 5 shows the block diagram for the adaptive filter method utilized in this work.

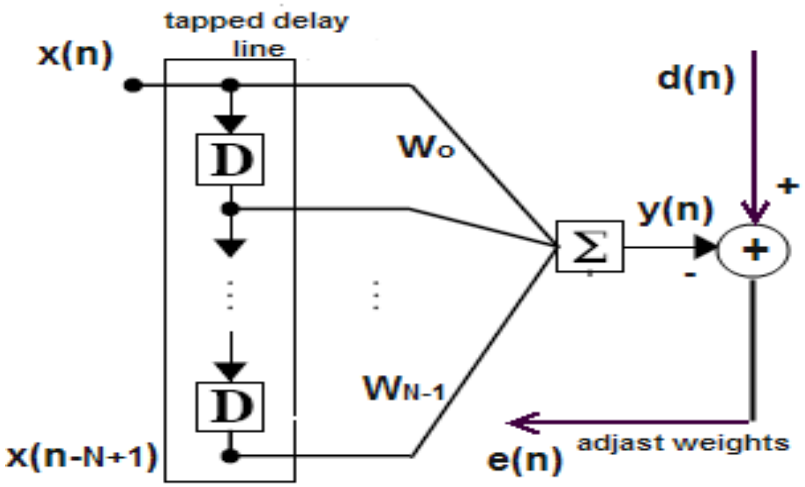

Figure (5) Adaptive filter block diagram .

Here w represents the coefficients of the adaptive filter tap weight vector, $x(n)$ is the input vector samples, the tapped delay line D, is needed to make full use of the filter. The input signal enters from the left and passes through N-1 delays. The output of the tapped delay line (TDL) is an N-dimensional vector, made up of the input signal at the current time, the previous input signal, etc. the $y(n)$ is the adaptive filter output, $d(n)$ is the desired echoed signal and $e(n)$ is the estimation error signal at time n.

The aim of an adaptive filter is to calculate the difference between the desired signal and the adaptive filter output, $e(n)$. This error signal is fed back into the adaptive filter and its coefficients are changed algorithmically in order to minimize a function of this difference, known as the mean square error function. In the case of acoustic echo cancellation, the optimal output of the adaptive filter is equal in value to the unwanted echoed signal. When the adaptive filter output is equal to desired signal the error signal goes to zero. In this situation the echoed signal would be completely cancelled and the far user would not hear any of their original speech returned to them, where the noise no(n) signal is assumed zero [2,8,11].

3.2 LMS Adaptive Algorithm

The least mean square error (LMS) algorithm is an example of supervised training, in which the learning rule is provided with a set of examples of desired network behavior. Here $\{x(0),d(0)\}\ \{x(1),d(1)\}$ ..... $\{x(N),d(N)\}$ is an input to the network, and is the corresponding target output. As each input is applied to the network, the network output is compared to the target. The LMS algorithm adjusts the weights of the filter so as to minimize this mean square error as shown equation(3.1).

$$mse = \frac{1}{N}\sum_{n=0}^{N-1} e(n)^2 = \frac{1}{N}\sum_{n=0}^{N-1} (d(n) - y(n))^2 \qquad \text{.....(3.1)}$$

Fortunately, the mean square error performance index for the linear network is a quadratic function called error surface. Thus, the performance index will either have one global minimum, a weak minimum, or no minimum, depending on the characteristics of the input vectors (see Fig. 6). Global minimum means convergence to optimum unique solution; where as weak or local minimums indicate the existence of many not optimum solutions. If there is no minimum, the performance diverge from optimum solution [11].

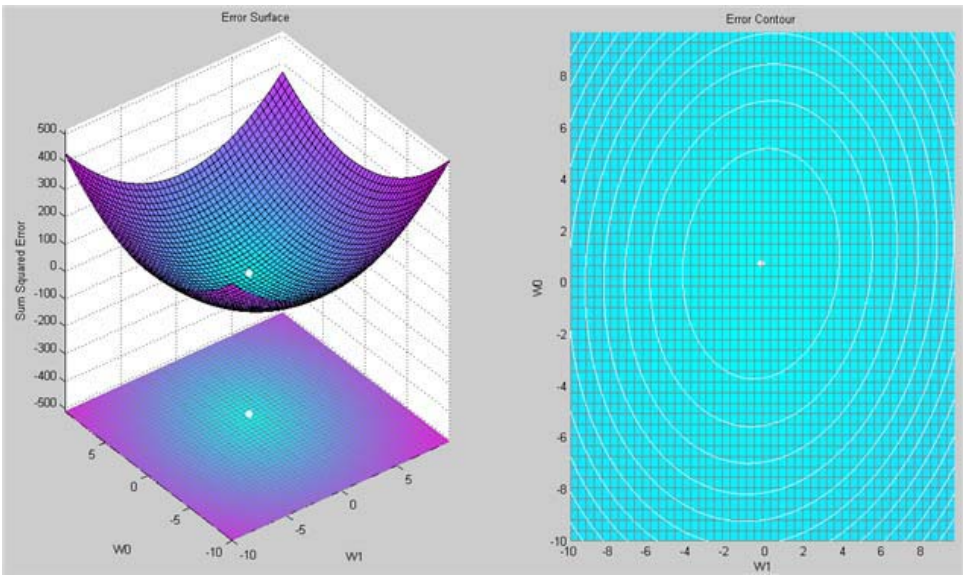

Figure (6) Error surface and error contour.

Adaptive networks will use the LMS algorithm or Widrow-Hoff learning algorithm based on an approximate steepest descent procedure. Here again, adaptive linear networks are trained on examples of correct behavior. The LMS algorithm, shown below, is discussed in detail [2,12]. The LMS algorithm is a gradient decent algorithms as it utilizes the gradient vector of the filter tap weights to converge on the optimal wiener solution. With each iteration of the LMS algorithm, the filter tap weights are updated according to the following formula shown:

$$\bar{w}(n+1) = \bar{w}(n) + 2\mu e(n)\bar{x}(n) \quad \text{..... (3.2)}$$

Where $\bar{x}(n)$ is the input vector of time delayed input values,

$\bar{x}(n) = [x(n)\ x(n-1)\ x(n-2).....x(n-N+1)]^T$.

The vector $\bar{w}(n) = [w_0(n)\, w_1(n)\, w_2(n).....w_{N-1}(n)]^T$ represents the coefficients of the adaptive FIR filter tap weight vector at time n. The parameter μ is known as the step size parameter and is a small positive constant. This step size parameter controls the influence of the updating factor. Selection of a suitable value for μ is imperative to the performance of the LMS algorithm, if the value is too small the time the adaptive filter takes to converge on the optimal solution will be too long; if μ is too large the adaptive filter becomes unstable and its output diverges.

The derivation of the LMS algorithm builds upon the theory of the wiener solution for the optimal filter tap weights, $\bar{w}_0$, as outlined in section above. It also depends on the steepest descent algorithm as stated in equation 3.4, this is a formula which updates the filter coefficients using the current tap weight vector and the current gradient of the desired function with respect to the filter tap weight coefficient vector, $\nabla\xi(n)$.

$$\bar{w}(n+1) = \bar{w}(n) - \mu \nabla \xi(n) \quad \text{.....(3.3)}$$

$$\nabla \xi(n) = E[e^2(n)] \quad \text{.....(3.4)}$$

As the negative gradient vector points in the direction of steepest descent for the N dimensional quadratic desired function, each recursion shifts the value of the filter coefficients closer toward their optimum value, which corresponds to the minimum achievable value of the desired function, $\xi(n)$. The LMS algorithm is a random process implementation of the steepest descent algorithm, from equation 3.4. Here the expectation for the error signal is not known so the instantaneous value is used as an estimate.

$$\xi(n) = e^2(n) \quad \text{.....(3.5)}$$

The gradient of the desired function, $\nabla \xi(n)$, can alternatively be expressed in the following form.

$$\xi(n) = \nabla(e^2(n)) = -2e(n)\bar{x}(n) \quad \text{.....(3.6)}$$

for the Implementation of the LMS algorithm. Each iteration of the LMS algorithm requires 3 distinct steps as shown in Table (1) below [2,4,11]:

Table (1) Implementation of adaptive LMS algorithm:

Step 1: The output of the FIR filter, $y(n)$) is calculated using equation 3.9.

$$y(n) = \sum_{i=0}^{N-1} w(n)x(n-i) = \bar{w}^T(n)\bar{x}(n) \quad (3.7)$$

step 2: The value of the error estimation is calculated using equation 3.10.

$$e(n) = d(n) - y(n) \quad (3.8)$$

step 3: The tap weights of the FIR vector are updated in preparation for the next iteration, by equation 3.11.

$$\bar{w}(n+1) = \bar{w}(n) - 2\mu e(n)\bar{x}(n) \quad (3.9)$$

Note that for each iteration the LMS algorithm requires 2N additions and 2N+1 multiplications (N for calculating the output, $y(n)$, one for $2\mu e(n)$ and an additional N for the scalar by vector multiplication).

4. Matlab Simulations

The adaptive filtering algorithm outlined in Section 3 was implemented using Matlab. The simulation of the echoed signal was generated by defining an appropriate impulse response then convolving this with a vocal input wav file. Figure 7 shows the desired signal, adaptive output signal, and estimation error signal for the LMS algorithm with vocal input, FIR filter order of 500 and step size of 0.01. The error signal shows that as the algorithm progresses the value of this signal decreases, this corresponds to the LMS filters impulse response converging to the actual impulse response. More accurately emulating the desired, signal more effectively canceling the echoed signal [2,12,13].

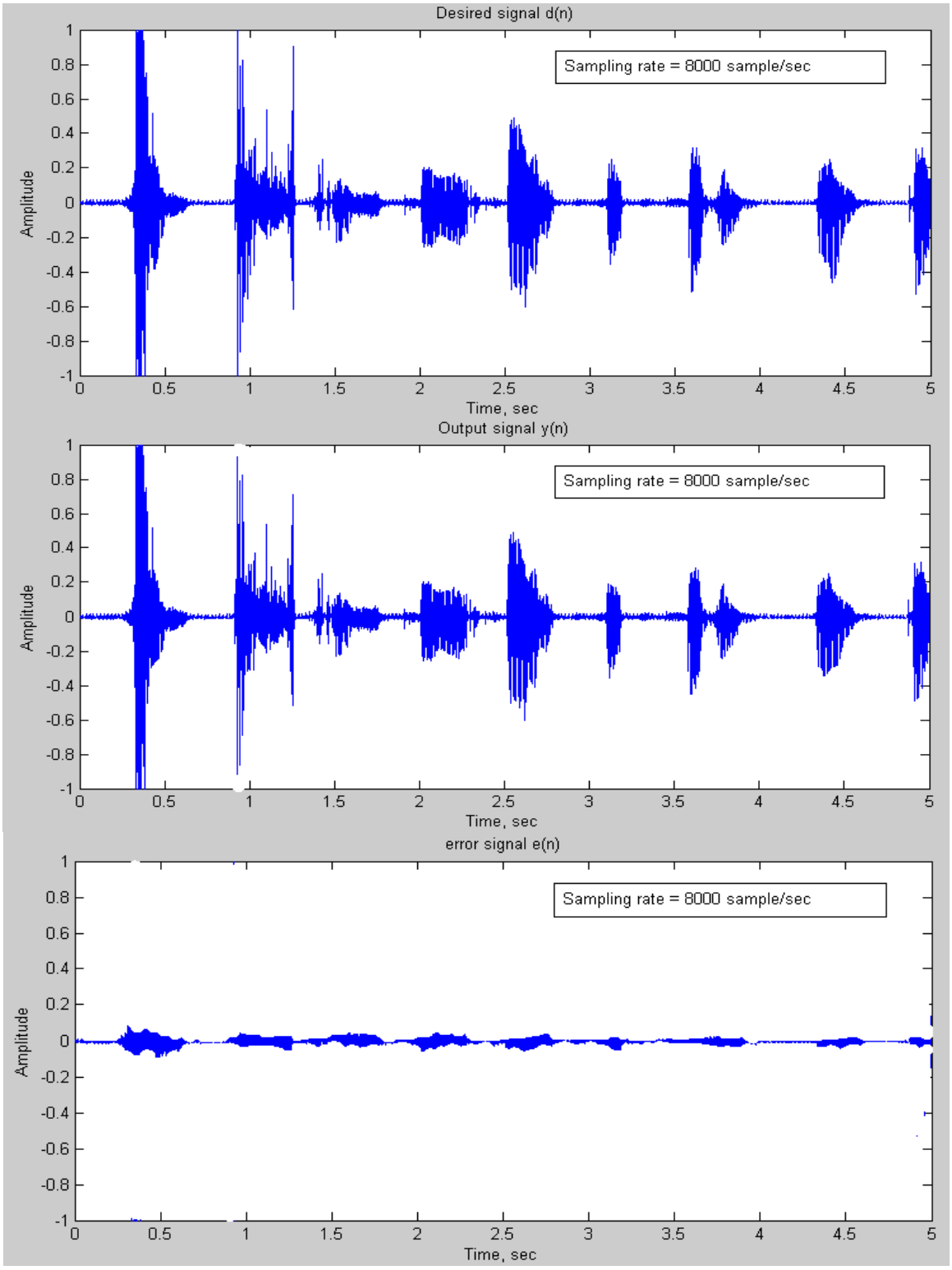

Figure (7) LMS algorithm outputs for vocal input.

A microphone is used to input a voice signal from the user, this is then fed in to the PC sound card, which is designed to acquire (through a microphone) and produce (through a speaker) acoustic signals , comes standard in almost every PC [9,10]. Here it is sampled by sound card at a fixed rate of 8 kHz. The first operation performed by the PC-based data acquisition system is a simulation of an echo response of a real acoustic environment. This is achieved

by storing each value of the sampled input in a file whose index is incremented with each new sample. The echoed signal is generated by adding the sample values and the stored values in the file (as a time delay image of original sample). This signal is then fed back into the file resulting in multiple echoes. The amount of time delay the echoed signal contains is determined by the number of sample stored in the file as given by following equation:

$$t_d = \frac{index\ length}{sample\ rate} = \frac{index\ length}{8000}$$

The file length (index) of the echo cancellation system can be adjusted 500, giving a time delay in the range of 50 ms. A round trip time delay simulates the effect of sound reflecting off an object approximately 10 meters away. This echoed signal is what the adaptive filter attempts to simulate. The better it does this, the more effective the cancellation will be. The new input is sampled and input to an array which represented by the input vector $\bar{x}(n)$. The step size value for this iteration of the LMS algorithm is then calculated. The output of the adaptive filter is then calculated by the dot product of the input vector and the current filter tap weight vector. This output is then subtracted from the desired signal to determine the estimation error value, $e(n)$. This error value, the step size value and the current FIR tap weight vector are then input to the LMS algorithm to calculate the filters tap weight to be used in the next iteration [2,4].

5. Hardware Implementation

Field Programmable Gate Arrays (FPGAs) can be reprogrammed as many times in order to achieve the desired result. The major design benefit in this lies in the ability to test designs that "might" work. Prior to the development of the FPGA, the fabrication process can be quite expensive and very time consuming. The use of FPGAs in the design process allows the more design flexibility, and reducing a cost and developing time. If the design fails after being tested on a FPGA, the designer can simply rework the design and download it again to the FPGA. Use of an FPGA would thus eliminate the loss in development time caused by a faulty initial design, as well as giving the designer knowledge of whether or not the design works [5].

5.1 Description of Spartan-3E Platform [14-15]

Figure 8 shows the Spartan-3E Starter Kit board, which includes the Xilinx Spartan-3E (XC3S500FT256) FPGA and other supported component. This Starter Kit , which is developed by Xilinx Inc., has been used as the hardware design platform for this paper.

The kit has many components that allows to develop and evaluate a design such as:

1- The Xilinx XC3S500E Spartan-3E FPGA has up to 232 user I/O pins, 320- pin FPGA package, and over 10,000 logic cells.

2- 4Mbit Flash Configuration PROM, 64 Mbyte DDR SDRAM

3- 50MHz on Board Clock source , Auxiliary Clock Oscillator Socket, and clock input/output Connector.

4- Four output ,SPI-based Digital-to-Analog Converter(DAC).

5- Two inputs based Analog-to-Digital Converter (ADC)with programmable gain pre amplifier.

6- 16 Mbits of SPI serial flash (STMicro).

7- 16 Mbyte (128Mbits)of parallel NOR Flash(Intel Stata Flash).

8- On- board USB-based FPGA/CPLD download/debug interface.

9-Programming of FPGA: the Spartan-3E Kit includes embedded USB-based programming logic and an USB endpoint with a Type B connector. Via a USB cable connection with the host PC, the iMPACT programming software directly programs the FPGA figure (8).

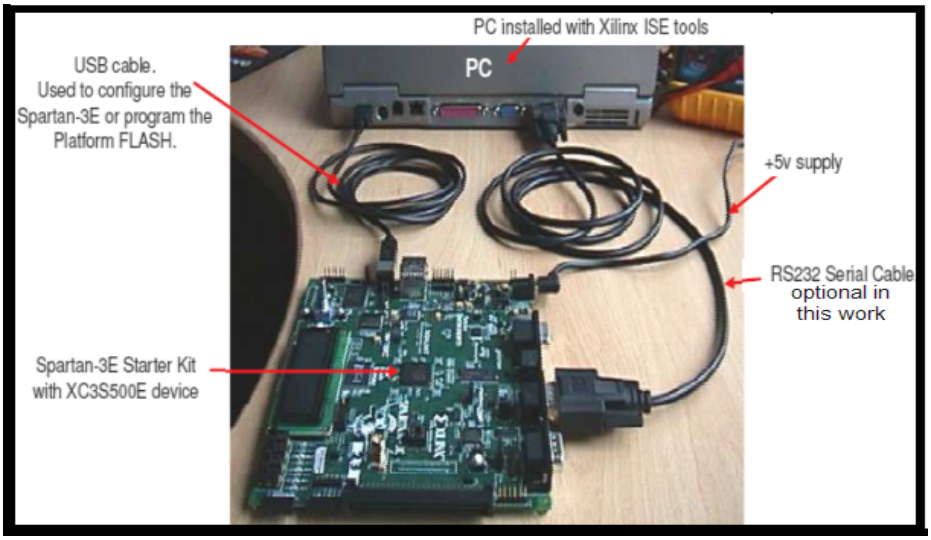

Figure (8)The equipments needed for hardware implementation.

5.2 VHDL Implementation of Adaptive Filter

The Register Transfer Level (RTL) of adaptive filter is shown in figure 9. The values of $\bar{x}(n)$ are stored in a shift register, whose outputs are connected to multipliers and then to adders. Also register are required for each weight coefficient. The structure is divided into N equal modules, called TAP0… TAPN-1. Each module (TAP) contains a slice of the shift registers, plus a multiplier and an adder. It also contains an output register, but this is optional (could be used at the last TAP only). This would increase the ripple propagation between the adders.

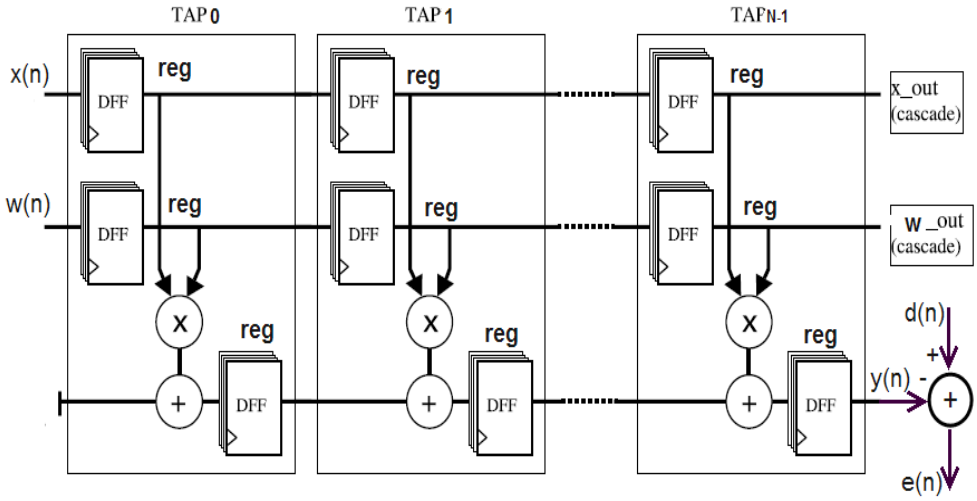

Figure (9) RTL representation of a adaptive filter.

The adaptive filter has coded in VHDL, simulated , synthesized with Xilinx ISE 8.2i tools, and has implemented in a Xilinx 'Spartan-3E Starter Kit board. The VHDL code for the adaptive filter in Figure 9 uses B bits to represent the input, weight coefficients, and reg. while 2B bits be used for all other signals (from the outputs of the multipliers all the way to y). notice that the lower section of the filter contains a MAC (multiply-accumulate) pipeline.

Here overflow can happen, so an add/truncate procedure must be included in the design. The time diagram results of adaptive filter as a *case study* is shown in figure 10. The values of weights were assumed as $w_0 = 4, w_1 = 3, w_2 = 2, w_3 = 1$.. With 4 bits to represent the inputs and weights, and 8 bits to represent the output, the synthesized circuit required 64 flip-flops (four + four + eight) for each stage of the shift registers.. Recall that the weights values are SIGNED (therefore, with 4 bit values, the range is from -8 to +7). The sequence applied to the input was $x(0) = 0, x(1) = 5, x(2) = -6,$ (16 – 6 = 10 in the graph). $x(3) = -1$ (16 – 1 = 15 in the graph), $x(4) = 4, x(5) = -7$ (16 -7= 9 in the graph), and $x(6) = -2$ (16 -2= 14 in the graph). Therefore, with all flip-flops previously reset, at the first positive edge of clk the expected output $y(0)$ is zero, which coincides with the first results for y in Fig. 10. At the next upward transition of clk, the expected value is $y(1) = w_0 * x(1) + w_1 * x(0) = 20$. And one clock cycle later, $y(1) = w_0 * x(2) + w_1 * x(1) + w_2 * x(0) = -9$ (256 -9 = 247 in the graph), and so on.

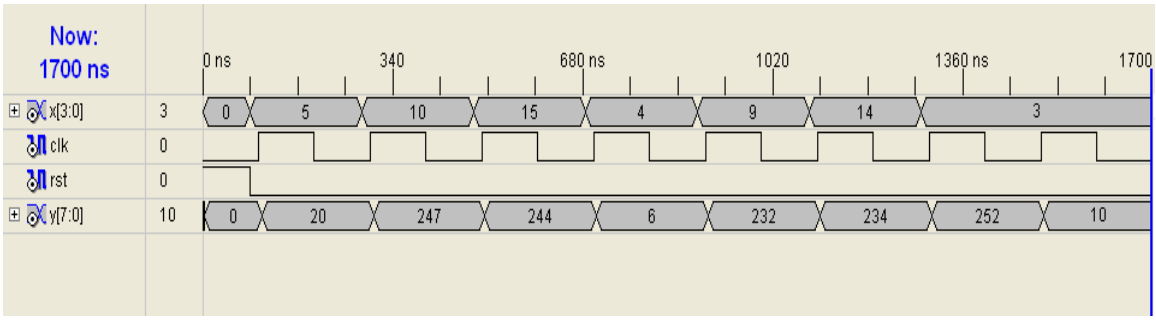

Figure (10) The time diagram of implementing an 8-bit /16-bit input/output 4 weight adaptive filter.

For an 8-bit input signal, 16-bit 0utput signal, and 8 weight coefficients with 8-bit for each weight. Figures 11 and 12, show the time diagram, and the RTL (register transfer level) hardware circuits for implementing adaptive filter.

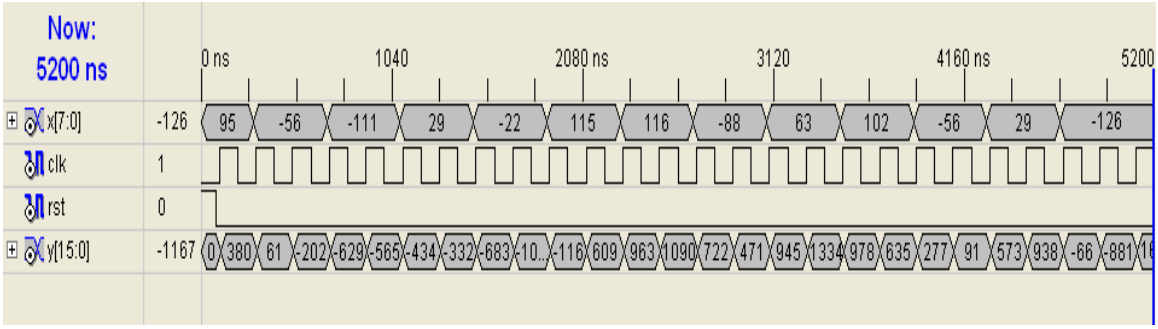

Figure (11) Time diagram of implementing an 8-bit /16-bit input/output adaptive filter.

The synthesized results is as shown in *Table (2)*, which gives performance and resource use summary for implementing the above adaptive filter. As it can be seen, the adaptive filter require very few hardware resources in comparison with the chip resources. The operation speed gives a good results and shows the advantages of using FPGAs in adaptive filter realization in comparison with software simulation. However, using a pure hardware implementation results in a much higher performance with some what lower flexibility. It shows a speed up close to 3.6x over the software implementation using Matlab performing on PC, with resources (Windows XP on 2GHz intel Centrino Duo processor, 2 MB cache, and 1GB RAM).

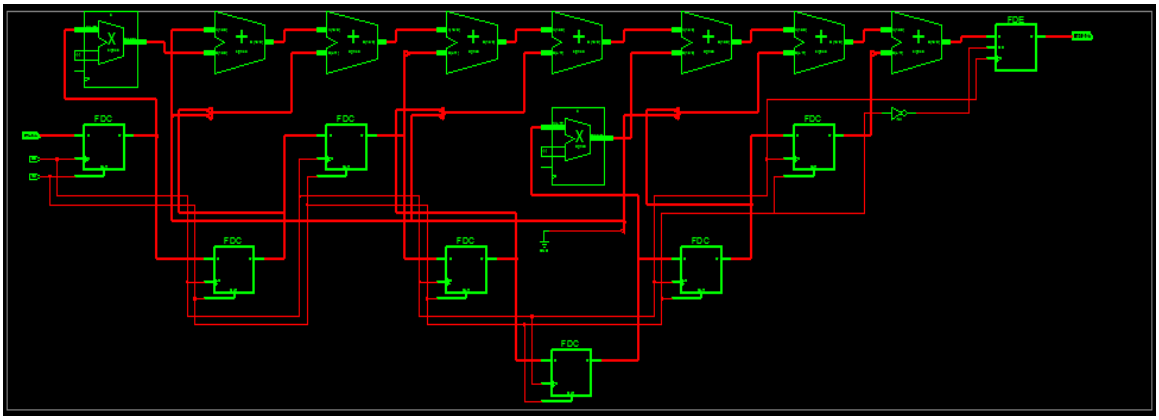

Figure (12) RTL hardware schematic circuit for implementing -bit /16-bit input/output adaptive filter.

Table (2) The synthesized results for implemented 8-bit/16-bit input/output adaptive filter

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-5

 Number of Slices:                         81  out of   4656     1%
 Number of Slice Flip Flops:               72  out of   9312     0%
 Number of 4 input LUTs:                  142  out of   9312     1%
 Number of IOs:                            26
 Number of bonded IOBs:                    26  out of    232    11%
 Number of MULT18X18SIOs:                   2  out of     20    10%
 Number of GCLKs:                           1  out of     24     4%

Timing Summary:
---------------
Speed Grade: -5

    Minimum period: 17.221ns (Maximum Frequency: 58.069MHz)
    Minimum input arrival time before clock: 14.012ns
    Maximum output required time after clock: 4.364ns
```

6. Conclusions

A conclusion of the performance of the LMS adaptive filtering algorithm is expressed by it's simplicity to implement, and it's stability when the step size parameter is selected appropriately. This made the LMS algorithm the acceptable choice for implementing acoustic echo cancellation system. Additionally, it does only require $2N+1$ multiplication operations. The acoustic echo cancellation system was successfully developed with the LMS algorithm. The system is capable of canceling echo with time delays of up to 50 ms, corresponding to reverberation off an object a maximum of 10 meters away. This proves quite satisfactory in simulating a medium to large size room.

There are many possibilities for further development in this work, This paper dealt with transversal FIR adaptive filters, this is only one of many methods of digital filtering. Other techniques such as infinite impulse response (IIR) or lattice filtering may prove to be more effective in an echo cancellation application, but with more hardware complications and need more than one FPGA chip to implement. The FPGAs constitute a very powerful option for implementing adaptive filter since we can really exploit their parallel processing capabilities.

References

1. Moonen, and Peroudeler," An Introduction to Adaptive Signal Processing", McGraw-Hill, second edition, 2000.
2. M, Hutson, "Acoustic echo cancellation using DSP", The School of Information Technology and Electrical Engineering, The University of Queensland, November 2003.
3. Widrow and Stearns, "Adaptive Signal Processing", Englewood Cliffs, NJ: Prentice Hall,1985.
4. Homer, J., Adaptive Echo Cancellation in Telecommunications. PhD thesis, University of Newcastle , Newcastle, 1994.
5. Chang Choo, " An Embedded Adaptive Filtering System on FPGA" Department of Electrical Engineering , San Jose State University, San Jose, CA 95198-0084, USA, 2007.
6. Robert Dony, et. al.," An FPGA Implementation of the LMS Adaptive Filter for Audio Processing", In Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM04), pages 324- 335, 2006.
7. U. Maser, L. Banbse, "Digital Signal Processing with Field Programmable Gate Arrays", International Signal Processing Conference, March 28, 2006, Dallas Texas.
8. Farhang-Boroujeny, B., "Adaptive Filters, Theory and Applications". John Wiley and Sons, New York, 1999.
9. Bellanger, Maurice G., " Adaptive Digital Filters", 2nd edition. Marcel Dekker Inc., New York, 2001.
10. Morris, Lu,"Adaptive Echo Cancellation",IEEE SIGNAL PROCESSING MAGAZINE, MARCH 1999.
11. "Neural network toolbox", Matlab 7.22, 2006.
12. Sibal, Sison, "Data acquisition of low- frequency data using a sound card", Instrumentation, Robotics, and Control Laboratory, University of the Philippines, 2004.
13. Hashmi, Butt, "Muti-channal data acquisition for implementation of real-time signal processing algorithm", Computer Engineering, Lahore University, Pakistan, 2005.
14. Xilinx , XST User Guide , Xilinx Inc., 2005.
15. Xilinx, Spartan™-3E Platform FPGAs Complete Data Sheet, Data Sheet DS031-4 (v2.0), Xilinx, Inc., 2006.

The work was carried out at the college of Engg. University of Mosul