FPGA Implementation of Adaptive Noise Canceller

Rafid Ahmed Khalil
Department of Mechatronics Engineering

Aws Hazim saber
Department of Electrical Engineering

College of Engineering, University of Mosul - Mosul, Iraq

Email: rafidamori@yahoo.com

Email: aws_anaz@yahoo.com

Abstract

This paper presents hardware implementation of least mean square (LMS) adaptive filter based Adaptive Noise Canceller (ANC) structure on FPGA using VHDL hardware description language. First, the adaptive parameters are obtained by simulating ANC on MATLAB. Second, the data, processed by FPGA, such as step size, input and output signals, desired signal, and coefficients of ANC, are exactly expressed into fixed-point data representation. Finally, the functions of FPGA-based system structure for such LMS algorithm in time sequence are synthesized, simulated, and implemented on Xilinx XC3S500E FPGA using Xilinx ISE 9.2i developing tool. The research results show that it is feasible to implement, on chip train, and use adaptive LMS filter based ANC in a single FPGA chip.

Keywords: *Adaptive noise canceller, least mean square, FPGA, Adaptive FIR filter*

تنفيذ نظام إلغاء الضوضاء المتكيف باستخدام مصفوفة البوابات المبرمجة حقليا

رافد احمد خليل
قسم هندسة الميكاترونيكس

اوس حازم صابر
قسم الهندسة الكهربائية

كلية الهندسة ,جامعة الموصل
موصل،عراق

الخلاصة

يعرض هذا البحث تنفيذ المرشح المتكيف المبني على خوارزمية معدل المربع الأقل (LMS) المستخدم في بنية نظام إلغاء الضوضاء المتكيف (ANC) على شريحة مصفوفة البوابات المبرمجة حقليا (FPGA) باستخدام لغة وصف الكيان المادي VHDL.أولا تم أيجاد العوامل المتكيفة لنظام ANC وذلك من خلال محاكاة نموذج ANC في البيئة البرمجية MATLAB. ثانيا إيجاد التمثيل المناسب لمعاملات المرشح و إشارات الإدخال و الإخراج باستخدام بيانات النقطة الثابتة و إيجاد كبر الخطوة المناسب لتنفيذ نظام ANC. و أخيرا تم التركيب النهائي لنظام ANC المبني على خوارزمية LMS ومحاكاتها و تنفيذها على شريحة Xilinx XC3S500E FPGA باستخدام أدوات التطوير الخاصة بشركة Xilinx ISE9.2i . أوضحت نتائج البحث إمكانية تنفيذ المرشح المتكيف المستخدم في ANC وتدريبه داخل الشريحة باستخدام شريحة FPGA واحدة.

Received 29 April 2008 **Accepted 3 Nov. 2008**

I. Introduction

Adaptive filters, as part of digital signal systems, have been widely used in communication industry, as well as in applications such as adaptive noise cancellation, adaptive beam forming, and channel equalization [1]. However, its implementation takes a great deal and becomes a very important field in digital system design. An adaptive filter is usually implemented in DSP processors because of their capability of performing fast floating-point arithmetic .But when FPGA (Field Programmable Logic Array) grows in area and provides a lot of facilities to the designers, it becomes an important competitor in the signal processing market. In addition, FPGA is a form of programmable logic, which offers flexibility for repetitive reconfiguration. Since FPGA consists of slices organized as array of rows and columns, a great deal of parallelism can be explored. Although it is not efficient to use floating-point arithmetic in FPGA due to its need for a large area ,it is sufficient to use fixed-point arithmetic for the adaptive filter to work well [3]. In general FIR structure has been used more successfully than IIR structure in adaptive filters [4]. The output FIR filters is the convolution of its input with its coefficients which have constant values. However, when the adaptive FIR filter was made this required appropriate algorithm to update the filter's coefficients [5].The algorithm used to update the filter coefficient is the Least Mean Square (LMS) algorithm which is known for its simplification, low computational complexity, and better performance in different running environments [6]. When compared to other algorithms used for implementing adaptive filters the LMS algorithm is seen to perform very well in terms of the number of iterations required for convergence. Recursive Least Squares algorithm, for example, is faster in convergence than the LMS but is then very complex to implement, hence detaining system performance in terms of speed and FPGA area used [7]. In order to use fixed-point arithmetic in adaptive FIR filters , the stalling phenomenon which may arise from fixed-point adaptation process should be avoided. This phenomenon can be achieved by a sufficient choice of bit length to represent the filter's coefficients [8].

This paper is organized as follows. Section II discusses the theory of adaptive FIR filters and adaptive noise canceller as well as the LMS algorithm. In section III the description of the implementation is given while section IV displays the results obtained. Finally section V gives the conclusions arrived at the results obtained.

II. Background Theory

A. Adaptive Filters

A filter is a device that maps its input signal to another output signal facilitating the extraction of the desired information contained in the input signal. Time-invariant filters have fixed internal parameters and structure. When specifications are given, the filter's transfer function and the structure defining the algorithm are fixed. An adaptive filter is time-varying since their parameters are continually changing in order to meet certain performance requirements. Usually the definition of the performance criterion requires the existence of a reference signal, which is absent in time-invariant filters. The general setup of an adaptive filtering environment is shown in Figure 1, where n is the iteration index, x(n) denotes the input signal, y(n) is the adaptive filter's output signal, and d(n) defines the reference or desired signal. The error signal e (n) is the difference between the desired d (n) and filter output y (n). The error signal is used as a feedback to the adaptation algorithm in order to determine the appropriate updating of the filter's coefficients. The minimization objective is for the adaptive filter's output signal matching the desired signal in some sense [7].

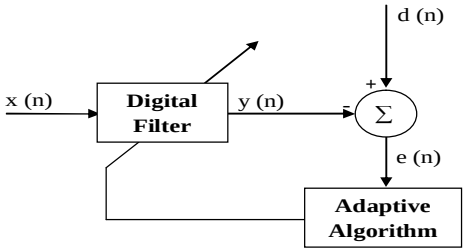

Figure 1 . Conventional Adaptive Filter Configuration

B. Adaptive noise cancellation

One of the adaptive filter applications is the adaptive noise canceller. Figure 2 describes its structure where the desired response is composed of a signal plus noise, which is uncorrelated with the signal. The filter input is a sequence of noise which is correlated with the noise in the desired signal. By using the LMS algorithm inside the adaptive filter, the error term e (n) produced by this system is then the original signal with the noise signal cancelled [9].

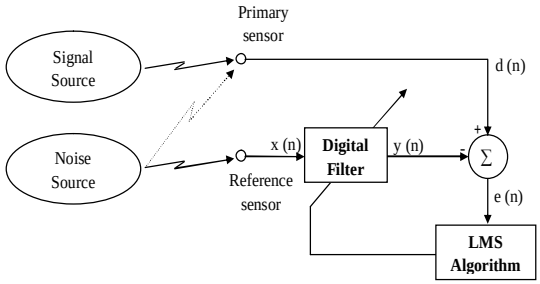

Figure 2 . Block diagram of adaptive noise cancellation

C. LMS algorithm

LMS is the most widely used algorithm. The key feature of the LMS algorithm is its simplicity. It requires neither measurement of the correlation function, nor matrix inversion [4] .It uses Mean Square Error (MSE) as a criterion. LMS uses a step-size parameter, input signal and the difference of desired signal and filter output signal to frequently calculate the update of the filter coefficients set [3].

1) LMS Equation: The simplest estimation may use only the current available taps and the current desired response to estimate the autocorrelation matrix and the cross-correlation vector. The equation to adapt tap weights w(n) using the instantaneous taps x(n) and desired response d(n) is [1]:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\, \mathbf{x}(n)[\, d(n) - \mathbf{x}(n)\mathbf{w}(n)\,] \qquad (1)$$

where μ is the step size, since the filter output is the convolution sum of the taps and tap weights

$$y(n) = \mathbf{x}(n)\mathbf{w}(n) \qquad (2)$$

and the estimated error signal e (n) is defined as the difference between the desired response and the filter response, or

$$e(n) = d(n) - y(n) \quad (3)$$

So, Eq. (1) can be rewritten in terms of the error signal and the taps:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\,\mathbf{x}(n)\,e(n) \quad (4)$$

Eq. (4) is the formula for the LMS algorithm [5]. As illustrated in the equation, each tap weight adaptation at each time interval requires merely the knowledge of the current taps and the current error signal, which is produced with the knowledge of the desired response. The algorithm does not require any prior knowledge of the entire autocorrelation matrix or the cross-correlation vector, nor does it require matrix computations.

2) The Convergence boundary: The convergence time of the LMS algorithm depends on the step size μ. If μ is small, then it may take a long convergence time and this may defeat the purpose of using an LMS filter. However if μ is too large, the algorithm may never converge. LMS algorithm can be shown to converge for values of μ less than the reciprocal of the largest eigenvalue of the autocorrelation matrix of x(n), but it may be time□varying, and to avoid computing it another criterion can be used [7]:

$$0 < \mu < \frac{1}{3LE[x^2(n)]} \quad (5)$$

where L is the number of filter taps and E[] represent the expected value of x2(n). The value of μ should be scientifically computed on the basis of the effects of the environment [7].

3) Selection of Adaptive Parameters: The choice of the step-size parameter and the order of the filter effectively determines the performance of LMS [3]. From Eq.(5) the range of μ is known but how can the value of μ be exactly chosen and how can the number of filter' taps be chosen? when the filter taps are increased, this improves the convergent performance of LMS algorithm, but every tap (in structure of LMS adaptive filter) costs two more multipliers and two more adders ,as seen in Figure 3. However, this will increase the area needed and decrease the maximum frequency of the design. So, balance is required between the convergent performance and the amount of hardware used effectively. Unfortunately, there is no clear mathematical analysis to derive the exact quantities. Only through experiments may a reasonable solution be obtained [3]. In order to select appropriate step size and filter order, MATLAB simulation of LMS algorithm is carried out. Based on the simulation results (which will be discussed later), the adaptive parameters obtained will be applied to the hardware implementation process of LMS algorithm.

III. Implementation

A. The Spartan-3E FPGA

The FPGA used to implement the adaptive filter is a 500,000-gate Xilinx Spartan-3E XC3S500E in a 320-ball Fine-Pitch Ball Grid Array package (XC3S500EFG320). Spartan-3E devices contain a two-dimensional row and column based architecture to implement custom logic. Its architecture consists of five fundamental programmable functional elements[10]:

- Input/output Blocks (IOBs)
- Configurable Logic Block (CLB) and Slice Resources
- Block RAM
- Dedicated Multipliers
- Digital Clock Managers (DCMs)

The Spartan-3E family features a rich network of traces that interconnect all five functional elements. Each functional element has an associated switch matrix that permits multiple connections to the routing [10]. Amongst the different architectures available for implementation, post testing results shows that direct convolution provided the fastest and least area consuming algorithm [7]. Testing showed that when the design is implemented in a modular approach, all the blocks are implemented as separate modules on the FPGA. The design is expensive in slice and flip-flop usage because of the additional components used in connecting the separate blocks together. This also results in increased data signal paths which carry with them increased delays and therefore lower the maximum possible frequency of the system operation. Therefore, in the optimization stage of the design the separate modules are combined together into one module using a single global clock [7].

B. Adaptive FIR filter Structural

The LMS algorithm uses an FIR filter structure. The design shown in Figure 3 represents a structural view of the filter. From the figure, the main components of the filter consist of L-1 Unit Delay Registers and L Weight Updates. The Unit Delay Registers are simply D Flip-Flops. Each Weight Update component consists of a multiplier, an adder and a buffer to store the new weights' update of the filter coefficient. According to equation (4) the filter output is subtracted from the desired signal to produce an error signal. The error signal is then multiplied with μ and then with the input signal, which produces next sets of filter coefficients.

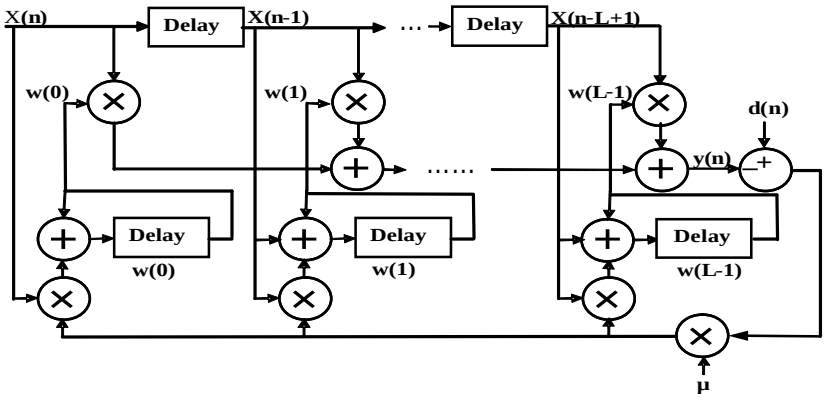

Figure 3 . Structural view of FIR Filter using LMS Algorithm

C. FPGA Implementation of LMS Algorithm

1) Processing of positive and negative number: In FPGA, the data, such as the input signals, the coefficients of filter, and the desired signal, may be positive or negative. So it is necessary to use signed numbers for expressing all the data inside FPGA. Based on the expression method of the signed number, the MSB bit is used as a sign bit. For the MSB bit, binary digit '0' denotes a positive number, binary digit '1' expresses a negative number, and all the data are denoted by the 2's complement.

2) Fixed-point representation of Data: Fixed point is a step between integer mathematics and floating-point. This has the advantage of being almost as fast as integer arithmetic, and able to represent numbers with fraction. It uses a smaller area in FPGA than floating-point to process the arithmetic operations. A fixed-point number has an assigned width and an assigned location for the decimal point. As long as the number is big enough to provide

enough precision, fixed point is fine for most DSP applications. Because it is based on integer math, it is extremely efficient as long as the data does not vary too much in magnitude.

The most essential task is the right selection of word lengths for the various variables in the system. Short word lengths may result in extra round-off errors, which can cause instability or poor performance. On the other hand, the use of excessively long word lengths increases system complexity which in turn reduces its maximum speed and increases the used area of the FPGA. So balance should be achieved between the system round-off errors and the maximum speed of operation together with the used area of the FPGA [8].

In order to select a sufficient word length and a range for the various variables in the system, we should design the proposed adaptive FIR filter according to the application (adaptive noise canceller) . So, a MATLAB simulation model of fixed-point adaptive noise canceller is used in order to get the variables' representation with minimum possible word length and sufficient partitioning between integer part and decimal fraction, which results from model tests with varying word lengths and decimal fractions [6]. A bad choice of a decimal fraction produces more quantization noise. Table 1 shows the numerical ranges of the input signals, the output signal, and the coefficients obtained by simulation. Based on the ranges obtained , the locations of the bits for the integer parts and those for the decimal fractions of each variable are obtained.

TABLE 1 . Numerical Ranges Of Variable

Variable name	***x***	***d***	***e***	***w***
Data range	***(-2,2)***	***(-2,2)***	***(-1,1)***	***(-0.2,0.2)***

For the input signal x range from -2 to 2 , its whole scale may use 2-bit, so the decimal fraction is located between the 9^{th} bit and the 1^{st} bit. According to this location method, the best precision can be obtained within the given dynamic range. Similarly, the decimal fraction of desired signal d is also located between the 9^{th} bit and the 1^{st} bit. So for other variables (such as y(n) and e(n)), the same location method is used. The selected number system is shown in Tables 2 and 3.

Table 2. Data format of x ,d & e

12	***11***	***10***	***9***	***8***	***7***	***6***	***5***	***4***	***3***	***2***	***1***
Sign	***Integer***		***Decimal fraction***								

Table 3. Data format of y

24	***23***	***22***	***21***	***20***	***19***	***18***	***…***	***13***	***12***	***11***	***…***	***1***
Sign	***Integer***				***Decimal fraction***					***Truncation***		

In selecting a proper word length for the fixed-point representation of the adaptive FIR filter tap weights, the stalling phenomenon is prevented. The stalling phenomenon comes from the bad selection of the fixed-point representation, the number of bits for the integer and that for the decimal fraction and the total word length (Figure 4 shows the simulation in ISE9.2i after implementation of adaptive FIR filter with the effect of stalling phenomenon when 8 bits is chosen as a word length). This means that the error signal does not approach the original

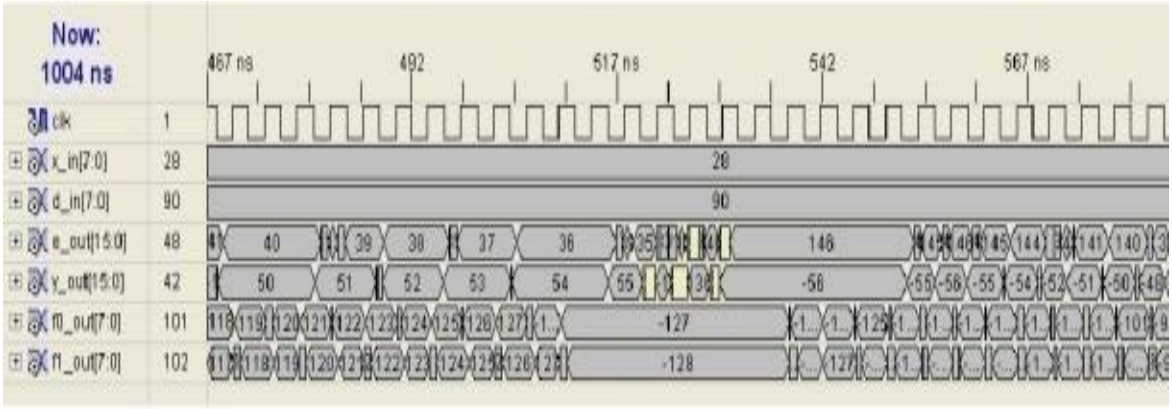

Figure 4 . Stalling phenomenon effect on adaptive FIR filter

signal and the adaptive filter's output will be unstable although the design will have smaller area and more speed as compared with the proposed design. The instability of the system happened when the weights passed the maximum number (in 8 bit representation with 2's complement is 127) at the adaptation time, which will cause the stalling phenomenon. Another source of this phenomenon comes from the small value of $\mu \mathbf{x}(n) e(n)$ so that the adaptation will be stopped[1].

So from the MATLAB simulation test of the proposed adaptive filter, the proper selection of adaptive filter weights for fixed-point representation is as shown in Table 4.

3) Step Size Selection Arithmetic: The step size parameter μ is a decimal number, and

Table 4. Data format of w

12	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*
Sign	***Integer***	***Decimal fraction***									

multiplying a decimal number is equivalent to dividing its reciprocal. However, in order to avoid division and multiplication operations for their consuming in maximum frequency and area for the design, Arithmetic Shift Right (ASR) operation is used instead of the division operation in order to increase the maximum frequency of the design and decrease its consumed area. The ASR operation on a 2's complement integer shifts the number n bits to the right (the direction of the least significant bit), while preserving the sign bit (the most significant bit). Shifting the number *n* bits to the right is equivalent to multiplying this number by $\mathbf{2^{-n}}$. Therefore, in order to achieve simplicity and feasibility, this design restricts the value of μ to be $\mu = \mathbf{2^{-n}}$, where *n* is a positive integer. From the MATLAB simulation test of the proposed design, μ equals 0 .004. ASR operation can provide us with $\mathbf{1/2^{8}}$ which is equal to 0.0039. So it seems that ASR is a very good choice for the proposed design.

IV. Testing And Results

The fixed-point LMS based adaptive filter was first tested using MATLAB simulation. Figure 5 shows the MATLAB model of fixed-point LMS based ANC, where the system shown in Figure 3 was implemented in software. Figure 6 shows the results obtained from Figure 5, confirming that the adaptive filter used performs well in obtaining the original signal from a

'noisy' version and gets peak signal to noise ratio improvement of 13.02 dB. From this model test, the minimum number of taps needed to get an acceptable filtered signal is 30 taps with a step size equal to 0.004 at word length and range as chosen before for implementation.
The hardware description language VHDL and FPGA are used to compile the source program of LMS algorithm. XC3S500E chip of Spartan-3E family from XILINX Corporation is selected, and the design is synthesized and simulated on ISE 9.2i. The simulation results after implementation of the fixed-point LMS based adaptive filter on FPGA are shown in Figure 7.
In Figure 7, x_in and d_in represent the inputs to the adaptive filter and the desired signal. From Figure 7 it is clear that the output signal is gradually stable after a period of time (time of convergence) and is close to the desired signal.f0_out & f1_out represent the first two coefficients in order to show what happened in the coefficient. This result denotes that the design can satisfy the desired demand and carry out adaptive process (the same method used in [1,6] to prove that the design achieves the adaptive process).

Device utilization summary:
Selected Device: 3s500efg320-4
Before map operation

Number of Slices:	4683	out of 4656	100%

After map operation

Number of Slices:	4654	out of 4656	99%
Number of Slice Flip Flops:	1560	out of 9312	16%
Number of 4 input LUTs:	8879	out of 9312	95%
Number of bonded IOBs:	100	out of 232	43%
Number of MULT18X18SIOs:	20	out of 20	100%
Number of GCLKs:	1	out of 24	4%

Timing Summary:
Speed Grade: -4
Minimum period: 37.164ns (Max. Frequency: 26.908MHz)
Minimum input arrival time before clock: 16.466ns
Maximum output required time after clock: 28.857ns

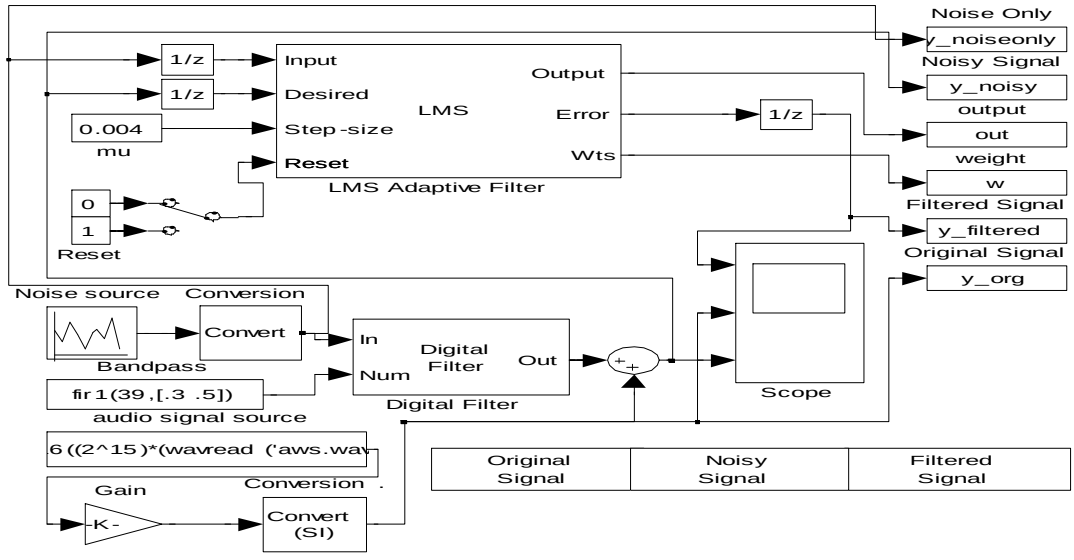

Figure 5 . MATLAB model of fixed-point LMS based ANC

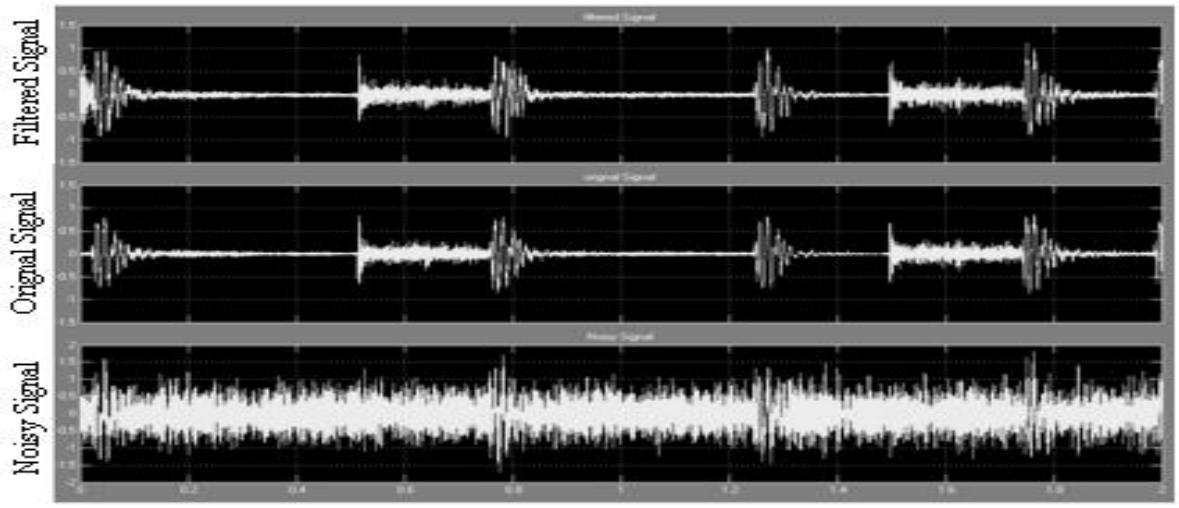

Figure 6 . Test Results obtained using MATLAB simulation

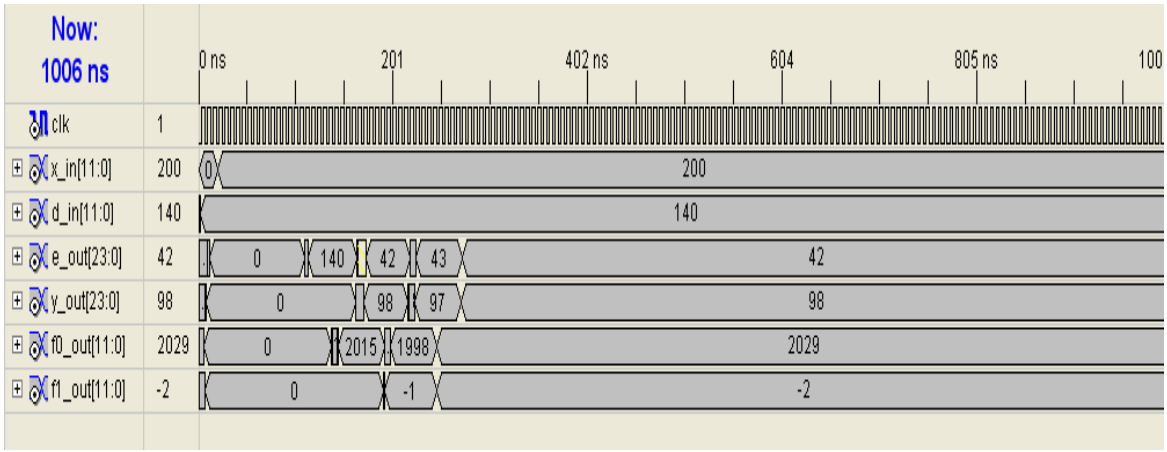

Figure 7 . Simulation result of the fixed-point LMS based adaptive FIR filter implementation on FPGA

V. Conclusion

The implementation of a fixed-point based adaptive FIR filter in a Spartan-3E device has been presented. The implementation is based on the design shown in Figure 5. The system worked as expected and the noise was cancelled out by the developed adaptive filter without any loss of data due to the bad selection of range or word length. The word-length requirement of various modules in the design has been discussed. It is found that although a relatively long word length should be used for the filter tap-weights to prevent the stalling phenomenon, the actual tap-weight bits, which should be used to calculate the filter output, can be many bits less.

References

[1] Simon Haykin, "Adaptive filter theory", Third edition, Prentice Hall, 2002.

[2] A. Elhossini, S. Areibi and R. Dony, "An FPGA implementation of the LMS adaptive Filter for audio processing," In Proceedings of IEEE International Conference on Reconfigurable Computing and FPGAs 2006 (RECONF'06), pp. 1–8, September 2006.

[3] A. Y. Lin, K. S. Gugel, and J. C. Principe, "Feasibility of fixed-point transversal adaptive filters in FPGA devices with embedded DSP blocks," In Proceedings of the 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications 2003, pp. 157–160, 30 June-2 July 2003.

[4] U. Meyer Baese, "Digital Signal Processing with Field Programmable Gate Arrays", Springer, 2006.

[5] B. Widrow and S. D. Steams, "Adaptive Signal Processing". Englewood Cliffs, NJ: Prentice-Hall, 1985.

[6] G. Yecai, H. Longqing, and Z. Yanping, "Design and implementation of adaptive equalizer based on FPGA," In Proceedings of IEEE 8th International Conference on Electronic Measurement and Instruments 2007 (ICEMI'07), pp. 4-790 – 4-794, August 16 2007-July 18 2007.

[7] M. Vella, and C. J. Debono, "The implementation of a high speed adaptive FIR filter on a field programmable gate array," In Proceedings of IEEE Electrotechnical Conference 2006 (MELECON'06), May 16-19, Benalmadena (Malaga), Spain, pp. 113–116, 16-19 May 2006.

[8] W. C. Chew, and B. Farhang-Boroujeny, "FPGA implementation of acoustic echo cancelling," In Proceedings of the IEEE Region 10 Conference 1999 (TENCON'99), Vol. 1, pp. 263–266, 15-17 September 1999.

[9] Woon-Seng Gan, Sen M. Kuo, Embedded Signal Processing with the Micro Signal Architecture, Wiley, 2007

[10] Xilinx, "Spartan-3E FPGA Family Complete Data Sheet", Data Sheet DS312, Xilinx, Inc.,2005.

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