less resources on FPGA, but maximum frequency is reduced 10 MHZ instead. Authors of [12] implemented the LMS adaptive filter on ALTERA Cyclone II FPGA. Their filter fastest convergence speed at $1/2^{10}$ step-size is 1.46 ms. Comparing this with the averaged convergence speed of the proposed method implemented on XILINX SPARTAN3E (200 us), shows 1460/200=7.3 speedup. And comparing with implementation on XILINX VIRTEX4 (125 us), shows 1460/125=11.68 speedup. Authors of [13] implemented LMS core for ANC system on TMS320-C40 DSP processor family. The computatio–nal time to execute on this DSP processor is about 6.2 ms. Comparing to proposed model implemented on SPARTAN3E FPGA, shows 6200/200=31 speedup. Authors of [14] implemented a real-time adaptive noise cancellation system based on an improved adaptive wiener filter on Texas Instrument TMS320C6713 DSK. They reported that floating point implementation of LMS filter takes worst case time of 38.95 ms to compute the filtering of heavy sine noisy signal consisting of 4096 samples per frame and the FIR filter is 40 taps long. Comparing with our proposed model with the same number of samples and filter taps implemented on SPARTAN3E FPGA, we have 38950/6553.6=5.94 speedup.

7- Conclusion

A fixed-point LMS-based adaptive noise cancellation core for FPGA-hardware implementation with low resource utilization was proposed. Some data problems were studied and effective methods were discussed. The core is based on FIR filter structure with 50 taps length and 16-bits signed coefficients. According to the results obtained, the system have successfully adapted and learned the environment statistics with a fast convergence speed. Comparisons with other implementations showed better convergence speed and lower resource utilization on FPGA. The proposed model is FPGA-brand independent so can be implemented on any FPGA brand (XILINX, ALTERA, and ACTEL). Although using a pure-hardware implementation results in high perform–ance than software or HW/SW co-design implementation, it is much more complex and low flexible. Future studies would be focused on implementing variable step-size NLMS algorithm (VSS-NLMS).

8- Acknowledgments

This work was supported by HESA Aviation Industries.

References

[1] S.M. Kuo, D.R. Morgan, "Active noise control, A tutorial review", IEEE Proc., Vol.78, pp.943-973, 1999.
[2] K.S. Chaitanya, P. Muralidhar, C.C. Rama Rao, "Implementation of reconfigurable adaptive filtering algorithms", Int. Conf. Sig. Proc. Sys., 2009.
[3] S. Haykin, "Adaptive filter theory", Prentice Hall, Upper Saddle River, NJ, 2002.
[4] M. Tarrab, A. Feuert, "Convergence and performance analysis of the normalized LMS algorithm with uncorrelated gaussian data", IEEE Trans. Info. Theo., Vol.34, No.4, pp.680-691, 1988.
[5] A.I. Sulyman, A. Zerguine, "Echo cancellation using a variable step-size NLMS algorithm", Conf. Sig. Proc. (EURASIP), 2004.
[6] Z.W. HU, Zhi-yuan XIE, "Modification of theoretical fixed-point LMS algorithm for implementation in hardware", 2nd Int. Symp. Elec. Comm. and Sec., 2009.
[7] T. Aboulnasr, "A robust variable step-size LMS-type algorithm: analysis and simulations", IEEE Trans. Sig. Proc., Vol.45, No.3, pp.631-639, 1997.
[8] W. Peng, B. Farhang-Boroujeny, "A new class of gradient adaptive step-size LMS algorithm", IEEE Trans. Sig. Proc., Vol.49, No.4, pp.805-810, 2001.
[9] K. Matsubara, K. Nishikawa, "A new pipelined architecture of the LMS algorithm without degradation of convergence characteristics", IEEE Conf. ICASSP, Vol.5, No.4, pp.4125-4128, 1997.
[10] H. Zheng-wei, X. Zhi-yuan, "Application of pipeline technique in realization of LMS algorithm", Jou. North China Elec. Pow. Univ., Vol.31, No.3, pp.93-96, 2004.
[11] V. Rodellar, A. Alvarez, E. Marinez, "FPGA implementation of an adaptive noise canceller for robust speech enhancement interfaces", 4th Southern Conf. Prog. Log., San Carlos de Bariloche, 2008.
[12] U.M. Mohdali, C. Umat, D. Alasadi, "Design and implementation of least mean square adaptive filter on altera cyclone II field programmable gate array for active noise control", IEEE Symp. Indu. Elec. App. (ISIEA), Kuala Lumpur, Malaysia, 2009.
[13] I. Majdar, M. Eshghi, "Implementation of PBS_LMS algorithm for adaptive filters on FPGA", IEEE Conf. TENCON, Chiang Mai, Thailand, 2004.
[14] X.S. Ganesan, M. Das, "Real time implementation of adaptive noise cancellation", IEEE Int. Conf. Elec./Info. Tech., No.4554341, pp.431-436, 2008.

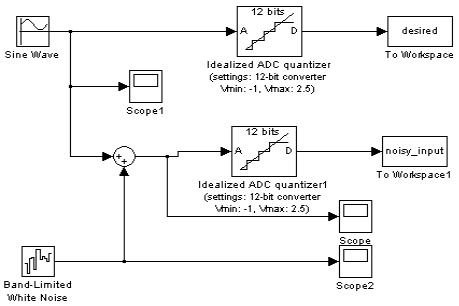


Fig. (5): Test data creation with band-limited white noise

It is consist of signal generators, 12bit ADC quantizer and etc. Test data is created with Simulink and then is exported to MATLAB workspace. An M-file is designed to write the results in separate text files to use in ModelSim VHDL simulator. Designed core VHDL file and Test Bench file is compiled in ModelSim Simulation software as VHDL93 standard. At the end of simulation the output data which has been written in a text file is brought to MATLAB workspace and converted from double to Int16. Then the required plots are derived.

5-2- Software Simulation Results

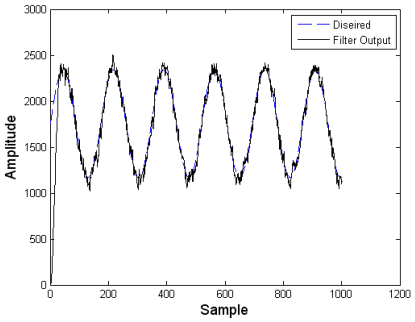


Fig. (6): (a). Desired and filter output signals

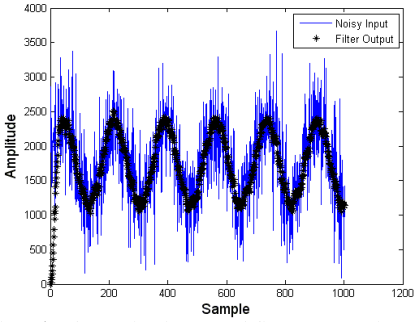


Fig. (6): (b). Noisy input and filter output signals

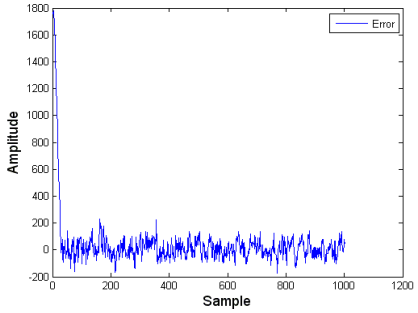


Fig. (6): (c) Residual error (learning curve)

In our simulation a 200 hertz sine signal is degraded with band limited white noise with 0.015 noise power (PSD). The results are shown in Fig. (6) SNR results are illustrated in table (3).

Table (3): Simulation performance

Input SNR	Output SNR	SNR Enhancement
1.0831 dB	8.7515 dB	7.6684 dB

5-3- Synthesis-Implementation Results

The proposed LMS-based ANC core is synthesized and implemented on XILINX VIRTEX5 (XC5vlx50t) chip using XILINX ISE software version 10.1. Resource utilization results are given in table (4) Some part of hardware-implementation result is shown in Fig. (7).

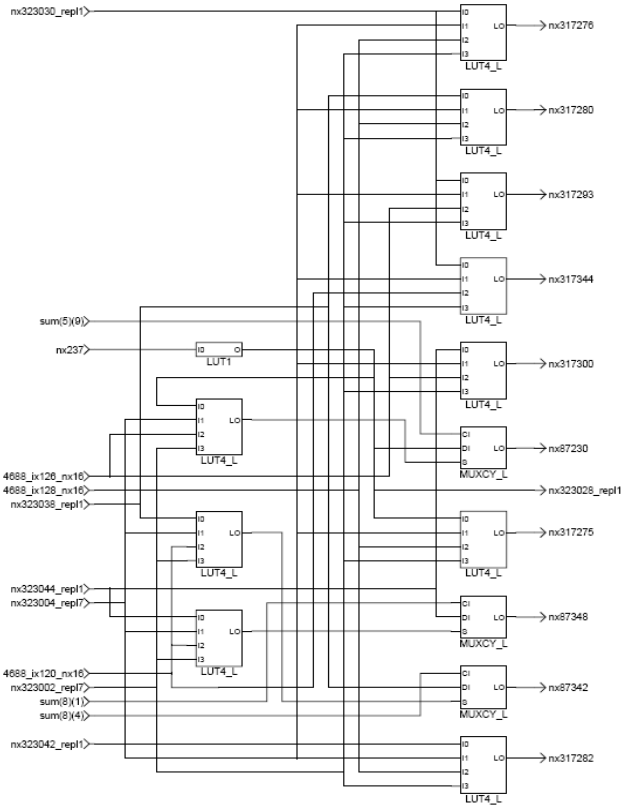


Fig. (7): Some part of implemented hardware on FPGA

Table (4): Implementation-resource utilization on Xc5vlx50t-3ff665

	Used	Available	Utilization
Slice Registers	2,618	28,800	9%
Slice LUTs	2,564	28,800	8%
occupied Slices	1,264	7,200	17%
bonded IOBs	54	360	15%
BUFG/BUFGCTRLs	1	32	3%
DSP48Es	2	48	4%
Maximum Frequency		103.581 MHz	

6- Comparison with Other Works

Authors of [11] implemented an adaptive noise canceller system on both XILINX (4VSX25ff668) and ALTERA (EP2S15F484C3) FPGA. Comparing the results shows that the proposed method in this paper utilizes about 16%

4-1- LMS Entity

The designed LMS entity is composed of reset, clock, adc_new_data, desired, input and output ports as illustrated in Fig. (2). Reset, clock and adc_new_data signals are one-wire ports while the others are 17 bits-long ports. The reset signal is used for resetting the algorithm and bringing it to initial values. The clock signal should be connected to processing clock. Because the algorithm should know when ADC has converted a new data, a signal called adc_new_data is provided so that it is asserted for a short time and then de-asserted. When working, first the output of the core is calculated, then is compared to the desired value and the error is obtained. The error is used to update the coefficients of the filter. This process is done repetitively. The ERD (Entity Relationship Diagram) of designed LMS-based FIR core is illustrated in Fig. (3).

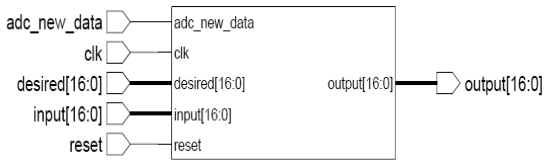


Fig. (2): Entity of fixed-point standard LMS core

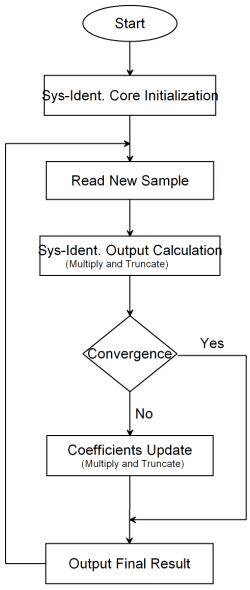
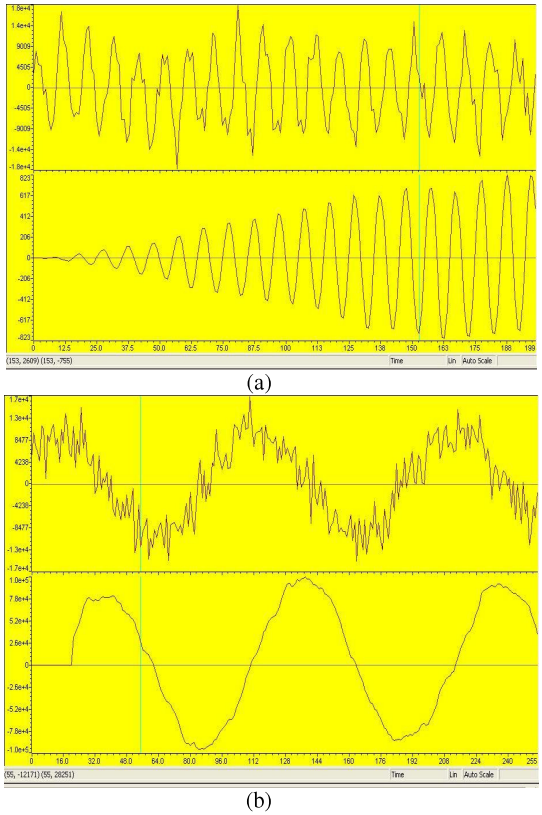


Fig. (3): ERD of LMS-based FIR core

5- Simulation of Proposed Model

The performance of the proposed FPGA-based adaptive noise canceller core is evaluated. But beforehand, the adaptive noise cancellation entity with LMS algorithm is simulated on TMS320C6416 DSP chip from TEXAS INSTRUMENTS Company. This processor works up to one Giga hertz in speed and thus can provide multiplica–tion and accumulation (MAC) in real time. This processor has two register banks called A and B, and also uses direct memory access (DMA) mechanism. Code composer studio (CCS) software is used for compilation and simulation. This software is capable of drawing graphs and calculating number of used machine cycles. In our simulations, the frequency of input signal in both tests is considered to be 100 and 10 hertz respectively. And additive white Gaussian noise (AWGN) is added to input signals. Preparation of input signals was done in MATLAB software and they were changed to 32-bit integer format to be used in CCS software. To enter data 3in CCS software, break point and probe point tools were used. Good performance of the designed core in reducing noise is shown in Fig. (4). By using clock profiling, the number of elapsed machine cycles for noise reduction of each input sample is 1509922 cycles. According to the processor clock which is one Giga hertz, the total time required to reduce noise from each input sample is 1.51 millisecond. Of course in the above situation, no optimiz–ation is done by compiler on the program code. If the optimization level is changed from the present situation (NONE) to FUNCTION(-O2) situation, the compiler will optimize the program code and reduce the number of machine cycles to 90118, which means the total time will be decreased to 90.12 microseconds, which is very low and good enough for real-time applications.

In the next section, the performance of the proposed FPGA-based adaptive noise canceller core is evaluated.



(a)

(b)

Fig. (4): Noisy input signal and cleared signal. Input signal frequency is 10 Hertz and SNR is 10 dB

5-1- Test Data Preparation

For simulation and verification of the designed FPGA-based ANC core, there should be some test data. MATLAB Simulink was used to create the essential test data as can be seen in Fig. (5).

Where M is the number of taps and S_{Max} is the maximum value of the power spectral density (PSD) of the tap inputs u. The good performance of the LMS algorithm and its simplicity has caused it to be the most widely used algorithm in practice. For an N tap filter, the number of operations has been reduced to 2N multiplications and N additions per coefficient update. This is suitable for real time applications and is the reason of LMS algorithm popularity [4].

Data should be converted into binary forms in order to be processed by digital systems. In fixed-point digital systems, data problems mainly involve: binary code representation, limited word-length selection, rounding and overflow. Rounding and overflow are correlated to limited word-length.

3-1- Binary code representation

There are several binary code representations. Performa–nces of one system based on different representations are different. Subtractions are included in LMS algorithm so results might be negative. For this reason, data should be represented as signed binary codes. The most popular representation of signed binary codes is two's comple–ment. Calculations of LMS algorithm are mainly done in decimals so decimals should be converted into integers firstly. When initializing weight vector values, initial values should be the values after conversion.

3-2- Limited word-length selection

In digital systems, every number can be represented by a binary code in limited word-length, so dynamic range and precision are finite. For LMS algorithm, the effect of limited word-length is that it will produce three errors: quantization errors of input vectors, quantization errors of weight vectors and quantization errors of calculation [5].

- Natural signals are analog signals that can not be processed by digital systems. In order to perform this task, analog signals should be converted into digital signals by using analog to digital converted into digital signals by using analog to digital converter (ADC). Samplings of the ADC are represented in limited word-length. There are differences between actual values and values of representations. These differences are quantize–tion errors of input vectors. Quantization errors can be reduced by improving the sampling precision of the ADC.
- Initial values of weight vectors also have to be represented by binary codes. Weight vectors are quantized according to the limited word-length. Quantiz–ation errors of weight vectors are produced in this process. Quantization errors of weight vectors can cause many problems such as actual results of filters deviate from theoretic results and so degrading the performance.
- Multiplication is one arithmetic operation in LMS algorithm. Rounding is needed in multiplication of two binary numbers with limited word-length. Two binary numbers that the length of each is N, then the length of the multiplication will be 2N. The length of the result should be rounding to N and N bit binary codes should be discarded. This were called calculation noise, this problem also is a quantization error. This noise can slow down the convergence speed, divergence of weight vectors and even lead the entire system to be collapsed. In order to make results more accurate, some measures can be taken into account. Appropriate algorithm structure can reduce the word-length effect and appropriate word-length can reduce the calculation noise. For impleme–ntation in hardware, long word-length numbers utilizes more resources than short word-length numbers. Perfor-mance and resource utilization should be balanced according to the requirement of the designed system.

3-3- Overflow

Errors can also be produced when overflow happens. These errors can slow down the convergence speed. Overflow can be avoided by two methods: extending the word-length of the accumulator and scaling data before calculation. The latter can be realized by shifting.

- Extending the word-length of the accumulator. When the word-length of input vectors and weight vectors is N, the number range that can be represented is $[-2^{N-1}, 2^{N-1}-1]$ if the two's complement is adopted. The bigger N, the range will be wider. But big value means more prices.
- Scaling can be realized by shifting operation. Because the binary codes are the two's complement, the sign bit should be settled appropriately. When left shifting, the sign bit is not changed, others are left shifted. Most significant bits are discarded and zeros are left shifted in. When right shifting, the least significant bits are discarded and sign bit is right shifted in. Final output results should be de-scaled, that is shifting in reversed direction [6].

Table (1): Input data bit-allocation

Sign Bit	Guard Bits	Word Length	Fraction Length
1	3	12	1

Table (2): Weights bit-allocation

Sign Bit	Word Length	Fraction Length
1	1	15

4- Fixed-Point Standard LMS Model

It was considered that we have a 12 bit binary output data from ADC unit, so the model for input data was designed as in table (1). The one bit fraction length is actually dummy and is not used for input/output data, but is necessary because of weight updates as we see then. For LMS Weights, the form in table (2) was considered. According to the LMS output formula:

$$Y=WU \tag{9}$$

The output of LMS (Y) would be 34 bit long. To be in format of 17 bits long, we truncate Y from 31 down to 15. The formula for updating weights is:

$$\Delta W = \eta EU \tag{10}$$

So the resulting length for W is 51 bits long. Then it is truncated from 33 down to 17 to fit in the desired format which is 17 bit long.

in reference [9]-[10]. This paper can be classified into the latter.

In this paper we first describe the theory of adaptive signal processing and LMS algorithms. Then in sections 3 and 4, data entry problems in LMS algorithm and description of designed fixed-point LMS-based adaptive noise cancellation core is given respectively. Section 5 shows simulation-implementation results and in section 6, a comparison is made with other works. At last, section 7 describes conclusions from the obtained results.

2- LMS algorithm

Adaptive filters learn the characteristics of their environment and continually adjust their parameters accordingly. Because of their ability to perform well in unknown environments and track statistical time variations, adaptive filters are employed in a wide area of fields. The adjustable parameters that are dependent on the applications are the number of filter taps, selection of FIR or IIR, choice of training algorithm, and the convergence speed (learning rate). Beyond these, the underlying architecture needed for realization is application independent. The main goal of any filter is to extract useful information from noisy data. Whereas a normal fixed filter is designed in advance with knowledge of the statistics of both the clear signal and the unwanted noise, the adaptive filter continually adjusts to a changing environment by the use of recursive algorithms [2,3]. This is useful when the characteristics of the signals are not known before of change with time.

The discrete adaptive filter in Fig. (1) receives an input X(n) and produces an output Y(n) by a convolution with the filters weights, w(k). A desired reference signal, D(n), is compared to the output to get an estimation error E(n). This error signal is used to incrementally adjust the filters coefficients for the next time instant. Several algorithms exist for weight update, such as the Least Mean Square (LMS) and the Recursive Least Squares (RLS) algori–thms. The selection of algorithm is dependent on required convergence speed and the computational complexity available, as statistics of the operating environment.

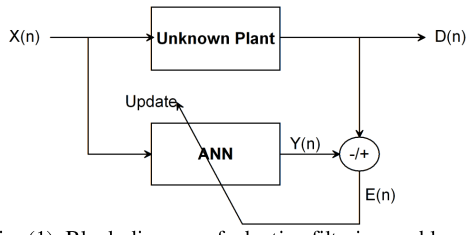


Fig. (1): Block diagram of adaptive filtering problem

2-1- Adaptive algorithms

There are some methods for performing weight update on an adaptive filter. There is the Wiener filter, which is the optimum linear filter in terms of mean squared error, and several algorithms that try to approximate it, such as the method of steepest descent. There is also least mean square algorithm for use in Artificial Neural Networks (ANN). Finally, there are other techniques such as the recursive least squares algorithm and the Kalman filter. The choice of algorithm is highly dependent on the signals of interest and the working environment, as well as the convergence speed required and computation complexity available.

2-2- Steepest descent method

With the error performance surface defined previously, one can use the steepest descent method to converge to the optimal filter weights for a given problem. Since the gradient of a surface (or hyper surface) points to the direction of maximum increase, then the opposite direction of the gradient ($-\Delta$) will point towards the minimum point of the surface. One can adaptively obtain the minimum by updating the weights at each time interval by using Eq. (1)

$$W_{n+1} = W_n + \eta(-\Delta_n) \quad (1)$$

In which the constant η is the step size (learning rate) parameter. The step size parameter determines how fast the algorithm converges to the optimal weights. A necessary and sufficient condition for the convergence or stability of the steepest descent algorithm is for η to satisfy Eq. (2)

$$0 < \eta < \frac{2}{\lambda_{Max}} \quad (2)$$

Where λ_{Max} is the biggest Eigen value of the correlation matrix R. Although it is still less complex than solving the Wiener-Hopf equation, the method of steepest descent is rarely used in practice because of high computation cost needed. Calculating the gradient at each time interval will include calculation of P and R, whereas the least mean square algorithm performs similarly using much less calculations.

2-3- Least mean square algorithm

The least mean square (LMS) algorithm is similar to method of steepest descent in that it updates the weights by iteratively approaching the MSE minimum. Widrow and Hoff invented this method in 1960 for use in neural network training. The key is that instead of calculating the gradient at every time interval, the LMS algorithm uses a rough approximation of the gradient. The error at the filter output can be expressed as Eq. (3)

$$e_n = d_n - W_n^T u_n \quad (3)$$

This is simply the desired output minus the filter output. By using this definition for the error, an approximation of the gradient is found by Eq. (4)

$$\Delta = -2e_n u_n \quad (4)$$

Substituting Eq. (4) for the gradient into the weight update Eq. (1) from steepest descent method gives Eq. (5)

$$W_{n+1} = W_n + 2\eta e_n u_n \quad (5)$$

This is the Widrow-Hoff LMS algorithm. As with the steepest descent algorithm, it can be shown to converge for values of η less than the reciprocal of λ_{Max}, but λ_{Max} may be time varying and to avoid computing it, another criterion Eq. (6) can be used.

$$0 < \eta < \frac{2}{MS_{Max}} \quad (6)$$

Hardware Implementation of LMS-Based Adaptive Noise Cancellation Core with Low Resource Utilization

Omid Sharifi Tehrani[(1)] – Mohsen Ashorian[(2)] – Payman Moallem[(3)]
(1) MSc. - HESA Aviation Industries
(2) Assistant Prof. - Department of Electrical Engineering, Islamic Azad University, Majlesi Branch
(3) Assistant Prof. - Department of Electrical Engineering, Isfahan University

Recive: Summer 2009 Accept: Summer 2010

Abstract: A hardware implementation of adaptive noise cancellation (ANC) core is proposed. Adaptive filters are widely used in different applications such as adaptive noise cancellation, prediction, equalization, inverse modeling and system identification. FIR adaptive filters are mostly used because of their low computation costs and their linear phase. Least mean squared algorithm (LMS) is used to train FIR adaptive filter weights. Advances in semiconductor technology especially in digital signal processors (DSP) and field programmable gate arrays (FPGA) with hundreds of mega hertz in speed, will allow digital designers to embed essential digital signal processing units in small chips. But designing a synthesizable core on an FPGA is not always as simple as DSP chips due to complexity and limitations of FPGAs. In this paper we design an LMS-based FIR adaptive filter for adaptive noise cancellation based on VHDL97 hardware description language (HDL) and Xilinx SPARTAN3E (XC3S500E) which utilizes low resources and is high performance and FPGA-brand independent so can be implemented on different FPGA brands (Xilinx, ALTERA, ACTEL). Simulations are done in MODELSIM and MATLAB and implementation is done with Xilinx ISE. Finally, result are compared with other papers for better judgment.

Index Terms: Adaptive noise cancellation core, FPGA, adaptive filters, hardware description language.

1- Introduction

One of the most important branches of signal processing is adaptive signal processing. The main purpose of adaptive signal processing is type of systems which are adaptively changed and self-adjusted. They can improve their perfor–mance by adaptively learning the characteristics and modifying their coefficients. Adaptive systems are widely used in many fields such as communication systems, channel estimation, equalization, sonar, radar, smart ant–enna, navigation systems, industrial control, prediction, system identification and active noise control (ANC) systems. Adaptive systems play an important role in their fields. In some cases especially where non-stationary and time-varying signals are concerned, the importance of adaptive signal processing is clear [1].

Recently requests for portable and embedded digital signal processing (DSP) systems have been increased dramatically. Applications such as audio devices, hearing aids, cell phones, active noise control systems with constraints such as speed, area and power consumption need an implementation by which these constraints are met with shortest time to market. Some possible solutions are ASIC chips, general purpose processor (GPP) and digital signal processor (DSP). Although the first option can provide a solution to meet hard constraints, it lacks the flexibility that is available in the two others. Using field programmable gate arrays (FPGAs) can reduce the gap between flexibility and high performance. New FPGAs include many primitives that provide DSP applications such as embedded multipliers, multiply and accumulate units (MAC), digital clock management (DCM), DSP-Blocks, and soft/hard processor cores (such as PPC). These facilities are embedded in FPGA fabric and optimized for high performance applications and low power consumption. The availability of soft/hard proc–essor cores in new FPGAs allows implementation of DSP algorithms without difficulty. An alternative choice is to move some parts of the algorithm into hardware (HW) to improve performance. This is called HW/SW co-design. This solution would result in a more efficient implement–tation as part of the algorithm is accelerated using HW while the flexibility is maintained. Another more efficient and more complex choice is to convert the whole algorithm into hardware as a pure HW implementtation. Although this is an attractive option under area, speed, performance and power consumption, the design will be much more complex [2]. Studies on LMS algorithm mainly concentrate on two aspects. One is the conver–gence time from the theoretical perspective; several mod–ified LMS algorithms were proposed in references [7]-[8]. The other is hardware implementation, in order to improve data throughput, many modified architectures for LMS algorithm such as pipeline technique were proposed