FPGA IMPLEMENTATION OF AN ACOUSTIC ECHO CANCELLER

Ioana HOMANA, Irina MURESAN, Marina TOPA, Cristian CONTAN
Technical University of Cluj-Napoca,ioana.homana@bel.utcluj.ro

Abstract: Acoustic echo cancellation (AEC) is an occurrence in today communication systems. The interferences caused by acoustic echo are distracting the users and reduce the quality of the voice. In order to attenuate the effects created by the echo, a Least Mean Square algorithm is used. In this paper, we present the implementation of an LMS adaptive filter, designed on a Xilinx Virtex-5 FPGA and implemented using VHDL. ModelSIM simulation results altogether with plots obtained in Matlab prove the behavior of the chosen adaptation algorithm for the specific application.

Keywords: acoustic echo cancellation, LMS, FPGA

I.INTRODUCTION

Acoustic echo cancellation (AEC) is one of the most popular applications for adaptive filters [1]. AEC devices are needed for removing the echoes resulting from the acoustic coupling between the loudspeaker(s) and the microphone(s) in communication systems (*Figure 1*). The main component of the system is the adaptive filter, which generates at its output a replica of the echo that is further subtracted from the microphone signal. One of the most used algorithms for AEC is the Least Mean Square (LMS) algorithm, due to its simplicity and low computational complexity. The update for the coefficient requires a small number of multiplications and additions, which makes it suitable for programmable gate array design. The advantage of using FPGAs is that allows applications to run in parallel or to be pipelined so that filtering, correlation and many other applications all run simultaneously.

In this work, the LMS adaptive algorithm is implemented on a Xilinx Virtex-5 FPGA development board, in order to show its performance in digital signal processing applications. A number of related publications exist on hardware design of the adaptive filters. In [2], FPGA implementation of an AEC based on variable step-size normalized least-mean-square (VSS-NLMS) algorithm is presented. Area and speed results are provided for a XC2S600E chip. Paper [3] proposes an FPGA implementation of an Adaptive Noise Canceller using the LMS algorithm. The hardware architecture is synthesized using the Xilinx Spartan-3e Starter Kit as the target board. It accelerates the filtering process and the speed up over pure software implementation increases with the filter tap.

The paper is organized as follows: section II gives the necessary background of the adaptive Least Mean Square algorithm and its application in the AEC system. Section III provides the detailed process of the FPGA implementation. Section IV presents the experimental results, in which the performance of the adaptive algorithm in the hardware architecture is shown. The last section summarizes the main findings of the paper.

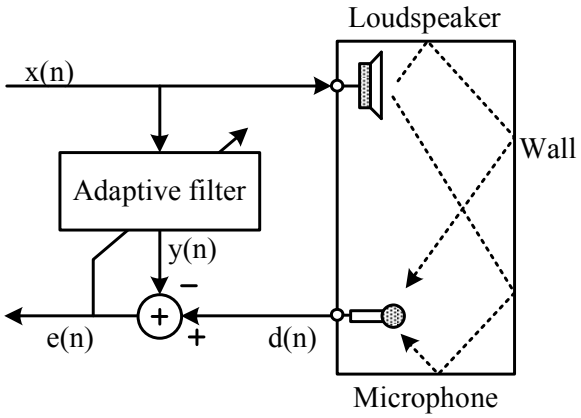

Figure 1.Basic schematic diagram of an AEC.

II APPLICATION OF LMS ALGORITHM IN AEC SYSTEM

Echo is the phenomenon in which a delayed and distorted version of an original sound or signal is reflected back to the source. The echo canceller must accurately estimate the echo path characteristic and rapidly adapt to its variation. This involves the selection of an adaptive filter and an algorithm for the adaptation. The best selection depends on the particular application and on performance requirements. In AEC scenarios, an adaptive filter identifies the acoustic echo path between the terminal's loudspeaker and microphone, i.e., the room impulse response.

Figure 1 shows the basic scheme of an AEC, where *x(n), y(n), d(n),* and *e(n)* are the input, output, microphone and error signals of the adaptive filter for time instant *n*. The input signal ***x****(n)* is filtered through the echo path to obtain the desired signal *d(n).* The error signal is obtained by subtracting the output signal from the microphone signal; its convergence proves the adaptation of the system.

The LMS algorithm is a stochastic gradient algorithm, which uses a fixed step-size parameter to control the updates for the tap weights of a transversal filter. Generally, the

Manuscript received May 6, 2011; revised June 22, 2011

objective function is to minimize the cost function [4]. The algorithm is based on two main processes:
a) filtering process, which involves,
- computing the output of a transversal filter produced by a set of input taps

$$y(n) = w_n^T(n) \cdot x(n), \quad (1)$$

where $(\,)^T$ denotes the transpose operation.
- generating an estimation error by comparing the output to a desired response

$$e(n) = d(n) - y(n). \quad (2)$$

b) adaptive process, which involves the automatic adjustment of the tap weights of the filter in accordance with the estimation error.

$$w_{n+1} = w_n + 2 \cdot \mu \cdot e(n) \cdot x(n), \quad (3)$$

where μ is the step-size parameter. In practice, to achieve a good tradeoff between fast convergence rate and low misadjustment, the value of the step-size has to be smaller than 1.

III FPGA IMPLEMENTATION

The Virtex-5 family contains five distinct platforms (sub-families), the best choice offered by any FPGA family. Each platform contains a different ratio of features to address the needs of a wide variety of advanced logic designs. Built on a 65-nm state-of-the-art copper process technology, Virtex-5 FPGAs are a programmable alternative to custom ASIC technology. Most advanced system designs require the programmable strength of FPGAs.

A Hardware design
The implementation of the LMS algorithm is based on the system block diagram from *Figure 2*. The System ACE compact Flash is a flash memory on which we can store the *.wav* signals and FPGA images. In order to control the memory flash read-write and to send the data to the MicroBlaze Soft IP Core Processor, a SysACE device is used. The MicroBlaze Processor saves this data in the RAM memory. When the data is saved in the RAM memory, the MicroBlaze Processor sends the samples of each signal, the input signal and the desired signal, to the adaptive filter, where the adaptation process takes place and the output signal of the adaptive filter is sent to the Audio Codec AC'97.

B VHDL implementation of adaptive algorithm
Figure 3 represents the digital block of the adaptive filter. The structure is divided into N equal modules, called TAP0… TAPN-1, each containing 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. Each module contains two shift registers. One of them stores the values of *x(n)*. The second shift register is required for each weight coefficient.

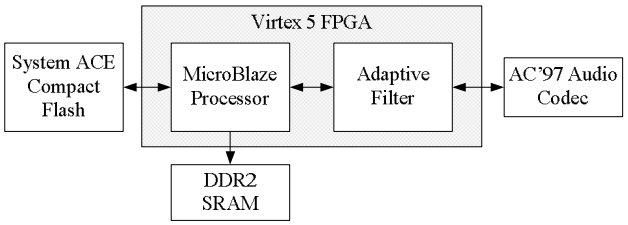

Figure 2. Logic Block Structure.

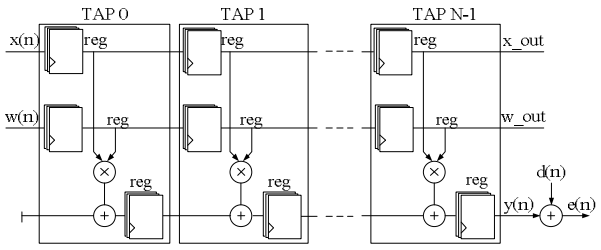

Figure 3. Digital block scheme of the adaptive filter.

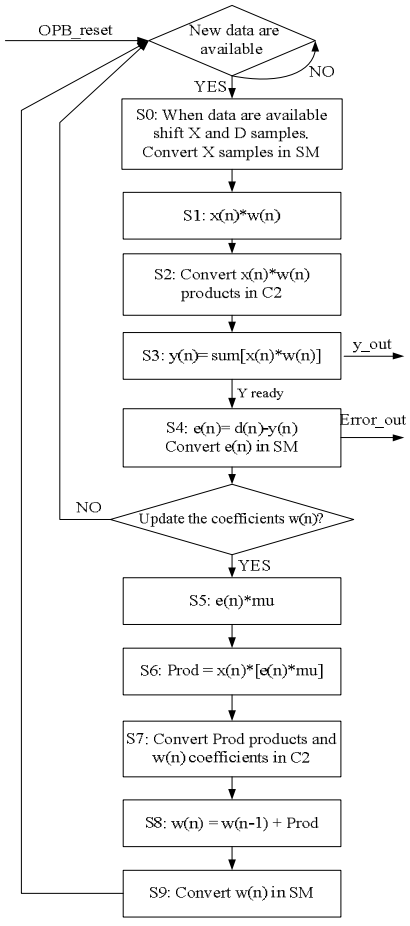

Figure 4. The LMS algorithm flowchart.

The output of these registers are connected to multipliers and then to the adders. The adaptive filter was coded in VHDL, simulated and synthesized with Xilinx ISE 12.4 tools. It has been implemented on a Xilinx Virtex-5 FPGA board.

C Step-by-step implementation of the LMS algorithm

Figure 4 represents de flowchart of the LMS algorithm, implemented using 9 states. S0 is the initial state of the system. When the data are available, the new data of the input signal *x(n)* and desired signal *d(n)* are shifted. The data received in S0 are in C2 (*two's complement*) form. In order to have a correct multiplication the data has to be converted in SM (*sign-and-magnitude*) and then forwarded in state S1, where the output signal *y(n)* from eq. (1) is computed. In S2 the products attained in the previous state are converted in C2, so that the addition can be done correctly. In the state S3 all products are added. Further, after all sums have been computed, the error signal *e(n)* of the LMS algorithm is calculated as in eq. (2) and the data are converted from C2 in SM.

We can observe that from S4 we can choose to cancel or not the adaptation process. If the adaptation process is cancelled, then we return in the initial state and wait for the new set of data to be available. Otherwise, if we do not want to stop the process, the adaptation will continue. Until now, only the equations of the input signal and the error signal were computed. Next, we will present the implementation adaptation technique used for the update of the adaptive filter coefficients, from eq. (3). In state S5, the product between the error signal *e(n)* and the step-size parameter μ, is made and then all products from the right side of the eq. (3) are computed in state S6. In S7 the coefficients of the adaptive filter *w(n)* and the product attained in S6 are converted in C2. The update formula of the adaptive filter coefficients is made in S8 and finally, in S9, the coefficients *w(n)* are converted in SM and sent back to the initial state, where the new set of data is expected to begin a new cycle of the adaptation process.

III SIMULATION RESULTS

This paper presents the implementation of an AEC on a Xilinx Virtex-5 FPGA development board. In this section the experimental results of the VHDL implementation of the LMS adaptive algorithm are presented. These results are worked out in the form of simulations and figures, showing the convergence of the Mean Square Error (MSE), this being the best proof of the system adaptation. Both ModelSIM and Matlab simulations were run, and one can see that the results are comparable in terms of MSE.

The adaptive algorithm used in AEC is the LMS algorithm. Simulations were performed using the ModelSIM 10 PE tools. The input signal *x(n)* is a speech signal of 10 seconds and can be seen in *Figure 6*. In Matlab, the speech signal is passed through a 256 taps acoustic impulse response (*Figure 5*), which represents the desired signal *d(n)*. The same length has been used for the adaptive filter. The input and the desired signals are processed in Maltlab. The data referring to these signals was transformed into specific files through a special conceived system. These files contain binary data, which characterize the signals at every clock cycle.

Since the convergence speed of LMS algorithm depends on the step-size μ, three different values are compared in the Matlab implementation (*Figure 7*). To measure the convergence speed the normalized misalignment (dB) is used, defined as $20\log_{10}(\|\mathbf{w}-\hat{\mathbf{w}}(n)\|_2 / \|\mathbf{w}\|_2)$, where $\|\cdot\|_2$ denotes the l_2 norm. The best result is obtained for μ =0.025, which is further used in both Matlab and ModelSIM simulation of the MSE.

In order to establish the performances of the LMS algorithm the MSE is used; it shows the adaptation curves of the algorithm employed (*Figure 8*). It is defined as the expectation of the norms of the square error as follow:

$$MSE = 10 \cdot \log_{10}\left(E\{\|e(n)\|^2\}\right) dB . \qquad (4)$$

In the digital design, after conceiving the VHDL test files, they were compiled and simulated in the ModelSIM environment. In order to visualize the MSE better than in ModelSIM, a special module was created in VHDL. It is instantiated in the top level and prints the signal into a file, which is loaded into a Matlab script and transformed into a plot (*Figure 9 and 11*).

A more objective measure to assess the echo cancellation by the adaptive filter is the Echo Return Loss Enhancement (ERLE) and is a discrete-time function, defined as the ratio of the instantaneous power of desired signal *d(n)* and the instantaneous power of the residual echo *e(n)*, defined as:

$$ERLE = 10 \cdot \log\left(\frac{P_d(n)}{P_e(n)}\right) dB, \qquad (5)$$

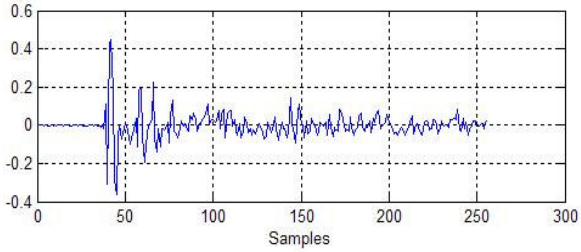

Figure 5. Acoustic Impulse Response

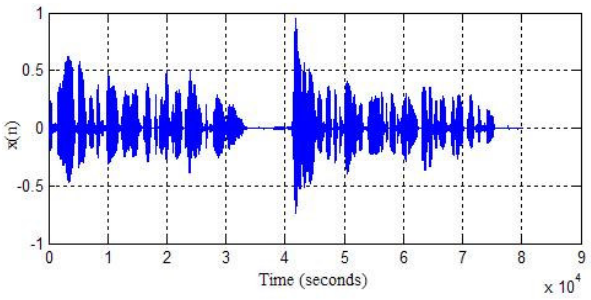

Figure 6. Input signal.

Logic Utilization	Used	Available	Utilization
Number of Slice Register	1241	149760	0%
Number of Slice LUTs	4032	149760	2%
Number of bonded IOBs	85	960	8%
Number of DSP48E	33	1056	3%

Table 1. Synthesis report.

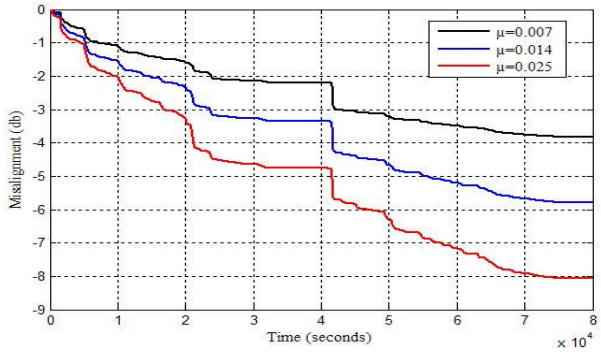

Figure 7. The convergence behavior of LMS algorithm for different step-size values.

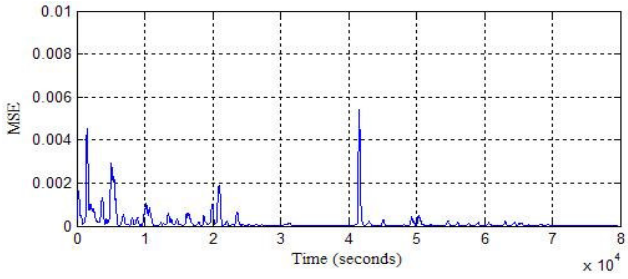

Figure 8. MSE convergence in Matlab environment.

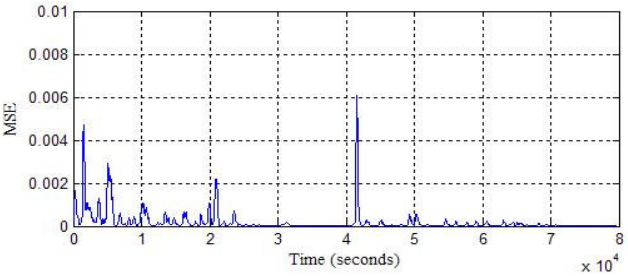

Figure 9. MSE convergence in ModelSIM environment.

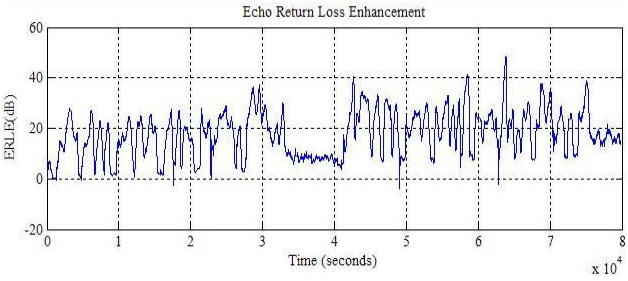

Figure 10. ERLE convergence in Matlab environment.

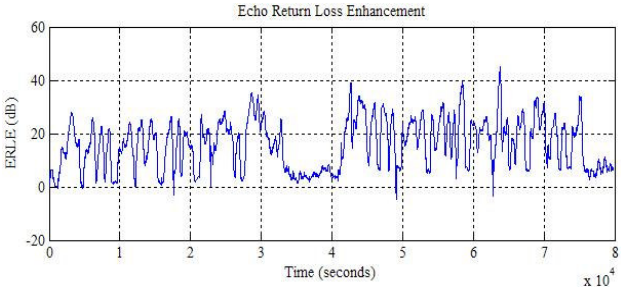

Figure 11. ERLE convergence in ModelSIM environmen.t

and makes a comparison of the echoes before and after cancellation (*Figure 10*). As in the case of MSE, the necessary signals to establish the ERLE performance, were loaded into a Matlab script and then transformed into plot.

The hardware usage reported by the synthesizer is given in Table 1. As we can see the logic utilization is low, which makes the LMS algorithm suitable for a real-time implementation.

IV CONCLUSIONS

A LMS adaptive filter implementation has been presented in this paper. The envisaged application is the acoustic echo cancellation. First the behavior of the LMS was studied in Matlab, where a filter of order 256 was programmed and simulated. Next a HDL design of an AEC was made, each component being created using the behavioral architecture of the VHDL hardware description language.

The input signal (speech) was provided from Matlab. This signal is transformed into a binary data file which characterizes it at each clock cycle. In order to verify the implementation, the test files together with the design file were compiled and simulated using ModelSIM environment. Plots regarding the convergence of LMS algorithm were provided using a special developed VHDL module that transforms data from ModelSIM into Matlab plots.

The experimental results confirm the good behavior of the algorithm obtained from the Matlab application. Summarizing the results, they indicate that it is feasible to implement adaptive filters in the digital domain. This code can be the source of programming a large area of FPGA devices because modern FPGAs contain many resources that support DSP applications, which are optimized for high performance and low power consumption. For a better precision, floating point operations should be performed in the hardware implementation.

A conclusion of the performance of the LMS adaptive filtering algorithm is expressed by its simplicity in implementation, and its stability when the step size parameter is selected appropriately. These advantages made the LMS algorithm the acceptable choice for implementing acoustic echo cancellation system on FPGA boards.

ACKNOWLEDGMENT

This paper was supported by Project development studies Ph.D. in advanced technologies "PRODOC" POSDRU/6/1.5/S/5 ID7676.

REFERENCES

[1] E. Hansler, G. Schmidt, *Acosutic Echo and Noise Control: A Practical Approach*, A John wiley & Sons, INC., Publication, 2004, ISBN 0471453463.

[2] Tian Lan, Jinlin Zhang, "*FPGA Implementation of an Adaptive Noise Canceller"*, IEEE 2008 International Symposiums of Information Processing, available on: http://doi.ieeecomputersociety.org/10.1109/ISIP.2008.107

[3] C. Anghel, C.Paleologu, J. Benesty, S. Ciochina, "*FPGA Implementation of an Acoustic Echo Canceller Using a VSS NLMS Algorithm",* Proceedings of IEEE ISSCS, 2009.

[4] R. Woods, J. McAllister, G. Lightbody, Y. Yi, *FPGA-based Implementation of Signal Processing Systems,* 2008 John Wiley and Sons