FPGA IMPLEMENTATION OF LMS AND NLMS ADAPTIVE FILTERS FOR ACOUSTIC ECHO CANCELLATION

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

Abstract: The paper proposes a register transfer level (RTL) description of two well-known adaptive algorithms used in acoustic echo cancellation: the Least Mean Square (LMS) and Normalized Least Mean Square (NLMS). The RTL descriptions of both algorithms are based on their finite state machine (FSM) diagrams and were created in the popular VHDL language. ModelSim simulation results altogether whit plots obtained in Matlab prove the good behavior. As it was expected, the performance of the NLMS algorithm is superior, compared to the LMS algorithm, in term of Mean Square Error (MSE).

Keywords: *VHDL description, adaptive filters, LMS algorithm, NLMS algorithm.*

I. INTRODUCTION

Multiple reflections in acoustic enclosures and transmission delay affect the sound quality, which in the case of a teleconferencing system lead to a poor understanding of the conversation. Public addressing systems are affected by acoustic feedback that may lead to the saturation of the system. In order to improve the sound quality and prevent audio feedback the acoustic echo cancellers (AECs) are deployed to remove the undesired echoes resulting from the acoustic coupling between the loudspeaker(s) and the microphone(s) [1].

The AECs is a system identification application (see Fig. 1), which uses adaptive filters to obtain a copy of the acoustic transfer function (the response of an enclosure to an acoustic impulse). The signal applied to the loudspeaker(s) *x(n)* propagates through multiple acoustic paths and it is picked up by the microphone(s). This signal is used as the desired signal *d(n)* in the system identification process. The output of the adaptive filter *y(n)* is obtained by convolving the samples *x(n)* with the adaptive filter coefficients *w(n)*. The filter is altered iteratively to minimize the error signal *e(n)*. The coefficient update can be carried out with many algorithms. One of the most popular adaptive algorithms available in the literature is the Least Mean Square (LMS) [2]. The main reason is the simplicity in implementation. Also widely used is the normalized version of the LMS algorithm, called Normalized Least Mean Square (NLMS) algorithm [3]. It exhibits a better balance between simplicity and performance than the LMS algorithm and has been used more often in real time applications. Both algorithms require a small number of multiplications and additions for the update of the coefficients, which make them suitable for digital design.

The growing use of Programmable Gate Arrays (FPGAs) in audio signal processing filters [4], [5], [6] determined the use of the popular VHDL hardware description language. The register transfer level (RTL) description is the only phase in the FPGA design flow that is carried out by human work; the other phases are done by automated design tools. In this paper, the LMS and the NLMS adaptive algorithms are described based on finite state machine (FSM) diagrams in order to study their performance in digital signal processing applications.

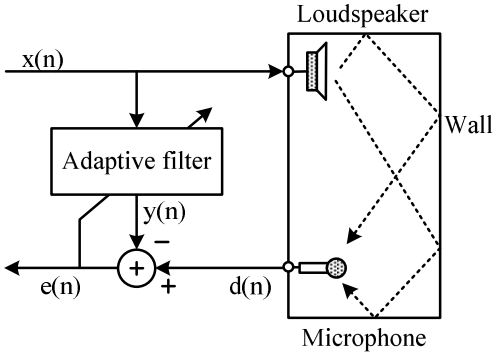

Figure 1. Basic layout of an AEC system

The paper is organized as follows: In Section II a brief theoretical description of the adaptive algorithms is given. Details on the FSM diagrams for LMS and NLMS algorithms are given in Section III. Section IV presents the behavioral simulation results of the RTL description, which is compared to MATLAB simulation results. In this way the functional verification of the RTL description is achieved. The last section summarizes the main findings of the paper.

II. ADAPTIVE ALGORITHMS

The block diagram of the adaptive filter from AEC system is represented in Fig. 2. Here w represents the coefficients of the adaptive filter tap weight vector, $x(n)$ is the input samples vector; the tapped delay line (TDL) D is needed to make full use of the filter. The input signal passes through

Manuscript received September 5, 2011; revised November 28, 2011

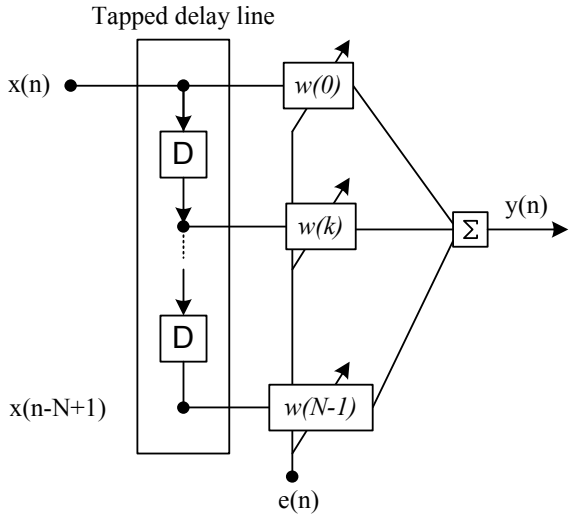

Figure 2. Adaptive filter block diagram

N-1 delays and the output of the TDL is a *N* length vector, made up of the input signal sample at the current time and the previous *N-1* samples. $e(n)$ is the estimation error signal at time n. An adaptive filter alters algorithmically its parameters in order to minimize the error $e(n)$, that is the difference between the desired echoed output $d(n)$ and its actual output $y(n)$:

$$e(n) = d(n) - y(n), \tag{1}$$

where the filter output is defined as follows:

$$y(n) = x^T(n) \cdot w(n). \tag{2}$$

The gradient descent algorithms (as LMS and NLSM) are aiming to minimize the mean square error (MSE). In each iteration, the error signal is fed back into the adaptive filter and its coefficients are changed algorithmically in order to minimize the MSE function. In the case of AEC, the output *y(n)* of the adaptive filter is equal to the desired signal *d(n)*, thus the error signal e(n) approaches 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 [7], [8].

The gradient (the direction of search for minimizing the MSE) of the LMS algorithm, is estimated with the current value of the error signal [9], so the filter tap weights are updated according to the following formula in each iteration:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2 \cdot \mu \cdot e(n) \cdot \mathbf{x}(n), \tag{3}$$

where μ is the step size (sometimes called learning rate) parameter. The step size is constant and does not depend on the input signal. This is a shortcoming of the LMS algorithm because a small step size LMS applied on a signal with large variance results in a long adaptation time.

The NLMS algorithm is an extended version of the conventional LMS; it employs the normalization of the input signal at each iteration and exhibits a good balance between simplicity and performance. The weight update of NLMS algorithm is expressed as:

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

where μ is defined as follows:

$$\mu(n) = \frac{1}{\delta + \mathbf{x}(n) \cdot \mathbf{x}^T(n)}, \tag{5}$$

and δ is a small positive constant, used to avoid the division by zero. The step size parameter controls the convergence speed and stability. For both algorithms the step-size parameter has to be positive $\mu > 0$, otherwise it will cause instability of the algorithm. Its range should be: $0 < \mu < (2/\lambda_{max})$, where λ_{max} is the largest variance of the input signal.

II. RTL DESCRIPTION

RTL description of digital circuits presents a special interest in the FPGA design flow because it is the only phase where human workload is involved. The RTL description was carried out using FSMs. The FSM diagrams of both algorithms are presented in the following.

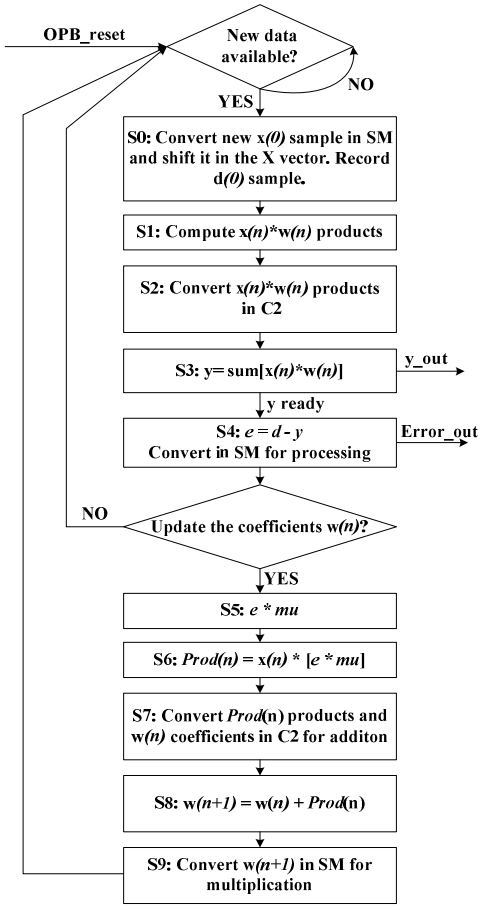

Figure 3. The finite state machine diagram of the LMS algorithm

A. VHDL implementation of the LMS algorithm

Fig. 3 represents the FSM diagram of the LMS algorithm, implemented using 9 states. S0 is the initial state of the system. When data is available, the new data samples set of the input signal $x(n)$ and desired signal $d(n)$ are shifted. The data received in S0 is expressed in C2 (*two's complement*) form. In order to have a correct multiplication, the data has to be converted in SM (*sign-and-magnitude*) form and then advanced in state S1, where the output signal $y(n)$ from eq. (2) 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. (1) and the data is converted from C2 in SM.

In state S4 we can choose to cancel or not the adaptation process. If the adaptation process is cancelled, then we return to 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 of the 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 achieved. 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 of the adaptive filter coefficients according to eq. (4) is done 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.

B. VHDL implementation of the NLMS algorithm

As we can see from the eq. (3) and (4), the difference between the LMS and the NLMS algorithms lies in the step size parameter, which implies an extra multiplication and a division. Fig. 4 represents the flowchart of the NLMS algorithm. If a new $x(0)$ sample is available, the new sample data of the input signal $x(n)$ is shifted in the x vector and the samples of the desired signal $d(n)$ are recorded. The sample data received in S0 is converted in SM and then forwarded to state S1. Next the output signal $y(n)$ from eq. (2) is computed and the new value of the product $x^2(n)$ is shifted in the N vector, where the norm $\mathbf{x}(n)$ is computed. In S2 the products of $x(n)*w(n)$ are converted in C2, so that the addition from the state S3 can be done correctly. There is no need to convert terms of the N vector in C2 because $x(n)*x(n)$ values are always positive. Also, in state S3 all the products of N vector are added. Further, the states S4, S5 and S6 are computed in the manner as in the LMS implementation.

In order to compute the quotient from the state S8, the initialization of the Prod(n) and Norm(n) values has to be done in the state S7. The update of the adaptive filter coefficients from eq. (4) is done in S10. Before the addition between the apriori adaptive filter coefficients $w(n\text{-}1)$ and the quotients, computed in S8, the C2 conversion of the quotients and the $w(n)$ coefficients takes place. Finally, in S11, 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.

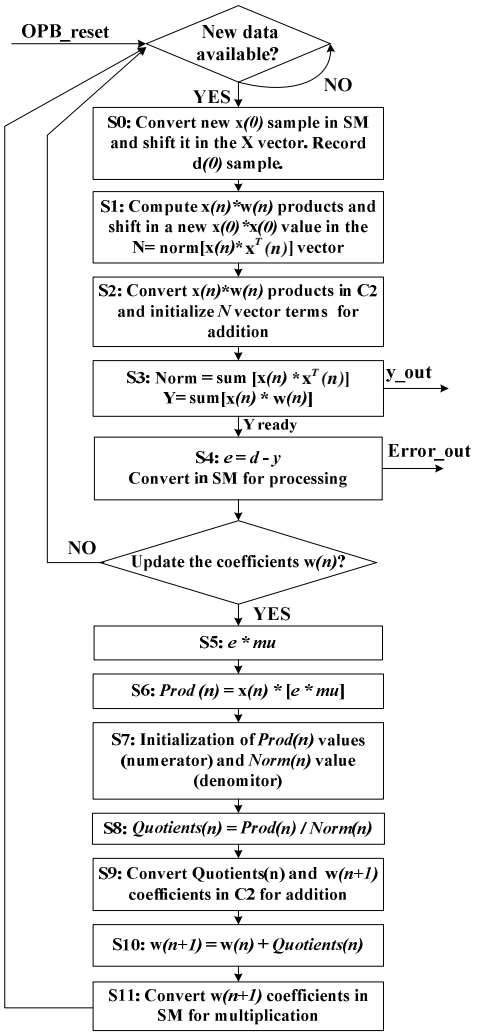

Figure 4. The finite state machine diagram of the NLMS algorithm

III. SIMULATION RESULTS

The verification of RTL descriptions is carried out by comparing behavioral simulations with MATLAB ones. The MSE plots are the best proof of the system adaptation, thus the MSE plots of he LMS and its normalized version, the NLMS are communicated in this work. We will demonstrate that the algorithms performance is comparable in both environments, so the RTL description can be considered correct and verified and it can be handed over to automated tools for FPGA implementation.

Behavioral 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 Fig. 5. In Matlab, the speech signal is passed through a 256 taps acoustic impulse response, 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 Matlab. 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. The value of the step size parameter was chosen to be 0,014.

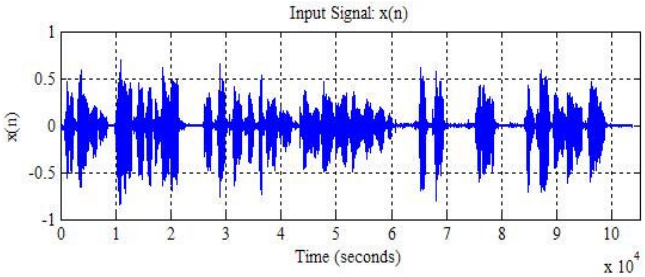

Figure 5. Input signal

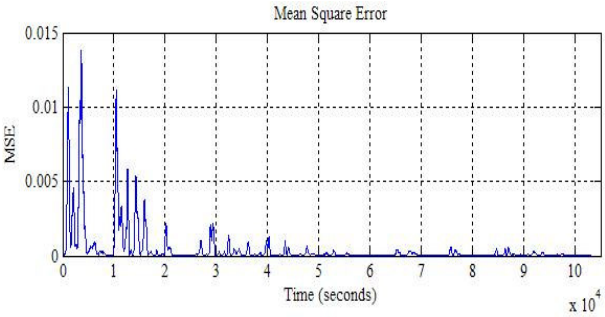

Figure 6. MSE convergence of LMS algorithm in Matlab

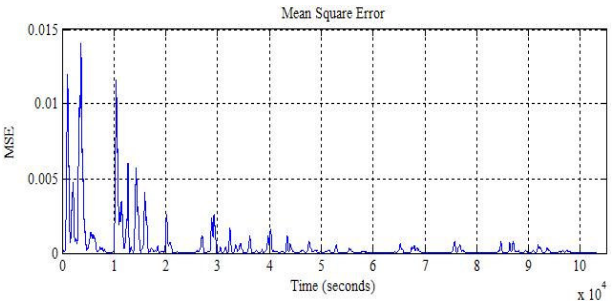

Figure 7. MSE convergence of LMS in ModelSIM

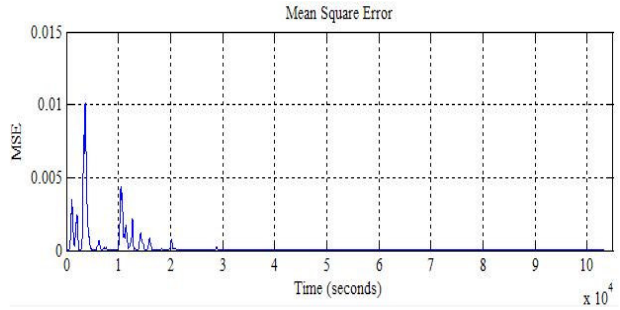

Figure 8. MSE convergence of NLMS in Matlab

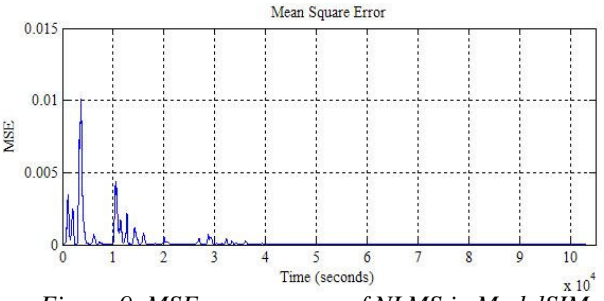

Figure 9. MSE convergence of NLMS in ModelSIM

The figure of merit used to measure the performance of the adaptive algorithms is the MSE. It is given mathematically as the expectation of the norms of the square error as follows:

$$MSE = 10 \cdot \log_{10}\left(E\left\{\|e(n)\|^2\right\}\right) dB. \quad (5)$$

Fig. 6 and 8 present the MSE performance of both algorithms in Matlab environment. Comparing the algorithms we can clearly see that the NLMS performs better than LMS algorithm, in terms of MSE.

In the digital design, after conceiving the VHDL test files for both algorithms, they were compiled and simulated in the ModelSIM environment. In order to visualize the MSE a special module was created in VHDL. It is instantiated in the top level and prints the signal into a file. For a better visualization of the results, the data are loaded into Matlab and plotted (Fig. 7 and 9).

IV. CONCLUSIONS

The RTL description based on FSM diagrams and comparison of two adaptive algorithms, the LMS and the NLMS algorithm are presented in this paper. First the adaptive algorithms were studied in Matlab, where a filter of order 256 was programmed and simulated. Next behavioral simulations of the RTL descriptions were carried out in ModelSIM. MATLAB and ModelSim simulation results are comparable, in this way the verification of RTL description was achieved. The results confirm that both algorithms obtained a good behavio ans, as expected, the NLMS algorithm outperforms the LMS algorithm, in terms of MSE. It has a better convergence. The NLMS adaptive filtering algorithm is expressed by its simplicity in implementation and its stability. These advantages recommend the NLMS algorithm as a good choice for real time implementation.

ACKNOWLEDGMENT

This work was sponsored by European Social Funds through "PRODOC" POSDRU/6/1.5/S/5 ID7676, "4D-POSTDOC" POSDRU/89/1.5/S/52603 and ID 2534.

REFERENCES

[1] J. Benesty, T. Gaensler, D. R. Morgan, M. M. Sondhi, and S. L. Gay, *Advances in Network and Acoustic Echo Cancellation*, Berlin, Germany: Springer-Verlag, 2001.

[2] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1985.

[3] S. Haykin, *Adaptive Filter Theory*, Fourth Edition, Prentice Hall, Upper Saddle River, N.J., 2002.

[4] Chang Choo, P. Padmanabhan, S. Mutsuddy, *An Embedded Adaptive Filtering System on FPGA,* Department of Electrical Engineering, San Jose State University, San Jose, CA 95198-0084, USA, 2007.

[5] R. Dony, et. al., "An FPGA Implementation of the LMS Adaptive Filter for Audio Processing", *Proc. of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM04)*, pp. 324- 335, 2006.

[6] Maser, L. Banbse, "Digital Signal Processing with Field Programmable Gate Arrays", *Proc. of the International Signal Processing Conference*, March 28, 2006, Dallas Texas.

[7] M, Hutson, *Acoustic echo cancellation using DSP*, The School of Information Technology and Electrical Engineering, The University of Queensland, November 2003.

[8] Farhang-Boroujeny, B., *Adaptive Filters, Theory and Applications,* John Wiley and Sons, New York, 1999.

[9] J. Benesty, C. Paleologu, T. Gansler and S. Ciochina, "A Class of Stochastic Adaptive Filters", *Signal Processing*, Vol. 4, pp. 29-48, 2011.