LABVIEW FPGA BASED NOISE CANCELLING USING THE LMS ADAPTIVE ALGORITHM

Erwin SZOPOS[1] Horia HEDESIU[2]
[1]*Bases of Electronics Department,* [2]*Electrical Machines, Marketing and Management Department, Technical University of Cluj-Napoca, Romania*
26-28 G. Baritiu str., Cluj-Napoca, Romania, Tel: +40264401803; Fax: +40264591340 *Erwin.Szopos@bel.utcluj.ro, Horia.Hedesiu@mae.utcluj.ro*

Abstract: This paper proposes an architecture for implementing the Least Mean Square (LMS) adaptive algorithm, using a 20 bit fixed-point arithmetic representation. The architecture length was established to 16, but it can be easily modified. This is an advantage for large filter orders. The method can also be applied to other LMS versions. This architecture is implemented using the NI cRIO-9104 FPGA chassis. The NI cRIO-9012 is a real-time module used for signal control and storage. Experiments were done regarding signal to noise ratio (SNR), filter length and type of input signals. The obtained results indicate the implemented algorithm as having high performance, while still incurring some limitations. The design is evaluated in terms of SNR, filter length and FPGA resources.

Keywords: *LMS, FPGA, SNR, real-time module, noise cancelling, filter length, step-size.*

I. INTRODUCTION

Filtering data in real-time requires dedicated hardware to meet demanding time requirements. If the statistical properties of the signal are not known, then adaptive filtering algorithms can be implemented to estimate the signals statistics iteratively.

DSPs and ASICs have traditionally been the common means for building and implementing adaptive filters. Due to the technological advance in the development of programmable logic devices, Field Programmable Gate Array (FPGA) has become attractive for realizing adaptive filters. FPGAs present 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 [1, 2]. However, many high-performance DSP systems, including LMS adaptive filters, may be implemented using FPGAs. Modern FPGAs include the resources needed to design efficient filtering structures. Furthermore, some manufacturers now include complete microprocessors within the FPGA fabric. This mix of hardware and embedded software on a single chip is ideal for fast filter structures with arithmetic intensive adaptive algorithms.

Adaptive filters learn the statistics of their operating environment and continually adjust their parameters accordingly. When the signal of interest and the noise reside in separate frequency bands, conventional linear filters are able to extract the desired signal [3]. However, when there is spectral overlap between the signal and noise, or the signal or interfering signal's statistics change with time, fixed coefficient filters are improper.

One of the most popular adaptive algorithms available in the literature is the stochastic gradient algorithm, also called the Least Mean Square (LMS) algorithm that is used in this work as well [4, 5]. Its attractiveness comes from the fact that it is very simple and robust. This adaptive algorithm is used to process a speech signal to enhance its signal to noise ratio (SNR). The algorithm is implemented on the National Instruments cRIO-9104 FPGA chassis. The acquired speech signal is read by the National Instrument cRIO-9012 real-time controller and then is transferred to the FPGA chassis for processing. A pure software architecture of the LMS algorithm was first proposed and tested using LabVIEW software [6, 7]. Finally a hardware architecture is mapped and tested using LabVIEW FPGA. The performance and area of the architecture is evaluated in terms of SNR, of the filter length, convergence speed, and FPGA resource usage [8].

This paper is organized as follows: section II deals with theoretical overview regarding the LMS algorithm and the processing modules used for implementation; section III details the implementation of the architecture; section IV presents the implementation results, and section V contains the conclusions of the paper.

II. THEORETICAL OVERVIEW

A. THE LMS ALGORITHM

The LMS algorithm was developed by Windrow and Hoff in 1959 as an example of supervised training where the learning rule is provided with a set of examples of desired behavior. Here the pairs $\{x[0],d[0]\}$ $\{x[1],d[1]\}$ ...$\{x[N-1]$,

Manuscript received June 26, 2009; revised August 2, 2009

$d[N-1]\}$ represent the input of the system; $d[n]$ is the corresponding target (Figure 1). As each input is applied to the network, the output is compared to the target. The LMS algorithm (Figure 2) adjusts the coefficients of the filter so that the mean square error is minimized [5]:

$$MSE = \frac{1}{N}\sum_{n=0}^{N-1} e^2[n] = \frac{1}{N}\sum_{n=0}^{N-1}\left[d[n] - y[n]\right]^2 \qquad (1)$$

where $e[n]$ is the error signal and $d[n]$ is the desired output signal. The LMS algorithm is a gradient descent algorithm as it uses the gradient vector of the filter coefficients to converge on the optimal Wiener solution. The filter coefficients are updated per iteration according to:

$$W[n] = W[n-1] + 2\mu \cdot e[n] \cdot X[n-1] \qquad (2)$$

where $\boldsymbol{W}[n] = [w_0\ w_1\ \ldots\ w_{L-1}]^T$ is the coefficients vector at time index k, $\boldsymbol{X}[n] = [x_n\ x_{n-1}\ \ldots\ x_{n-L+1}]^T$ is the data vector of the L most recent input samples and μ is the convergence factor or step-size. This factor controls the stability and the convergence rate of the adaptive algorithm [8, 9]. The output of the LMS filter is:

$$Y[n] = W[n-1]^T \cdot X[n] \qquad (3)$$

B. THE USED HARDWARE

The National Instruments cRIO-9104 reconfigurable embedded chassis [9] is used to implement the LMS hardware architecture. This chassis contains the Virtex-II FPGA chip with 3M gates that offers ultimate processing power and the ability to design custom hardware using LabVIEW FPGA software. The cRIO-9014 is used for audio signal controlling/storing and for configuring the FPGA chassis. This is an embedded real-time controller module featuring an industrial 400MHz Freescale MPC5200 real-time processor for deterministic and reliable real-time applications. These two modules provide a complete platform to implement audio applications based on Xilinx FPGAs. The chassis containing the FPGA has 8 slots to connect external hardware modules, used for signal acquisition. The modules are interfaced to the FPGA in order to enable transferring data directly to the chip. The real-time controller contains a 10/100BaseT/TX Ethernet port to communicate with the PC or with other systems. The LabVIEW suite is used to implement the hardware architectures for signal processing.

III. FPGA IMPLEMENTATION

Figure 1 shows an adaptive filter in the role of noise canceller [10]. The reference input $x[n]$ is the noisy signal applied to the adaptive filter and the primary input $d[n]$ is the noise signal or is a highly correlated version of it. By subtracting the reference signal from the adaptive filter's output the error signal $e[n]$ is obtained (equation 1) that is used to compute the filter coefficients. The aim of the adaptation algorithm is to adjust the filter coefficients such that the filter output $y[n]$ is as close to the speech signal as possible in some mean or average sense.

Figure 2 shows the block diagram of the LMS algorithm to compute only one coefficient at instant k.

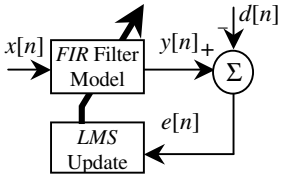

Figure 1. Adaptive noise cancelling block diagram.

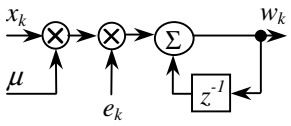

Figure 2. Block diagram of LMS.

Figure 3 shows the LMS architecture implemented in LabVIEW FPGA to compute a *16* coefficients FIR filter.

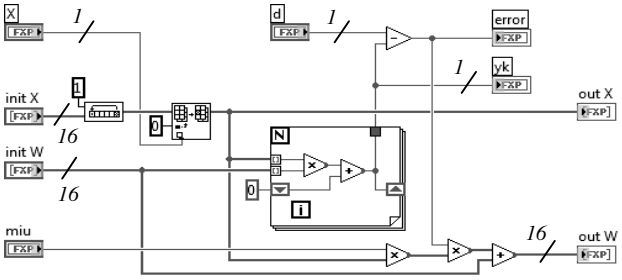

Figure 3. LabVIEW FPGA architecture of the LMS algorithm.

One of the tasks performed is to shuffle the past samples of the contents in the vector $\boldsymbol{X}[n]$. The two array functions *Replace Array Subset* and *Rotate 1D Array* act as a circular buffer where the input sample at index *0* gets replaced by a new incoming sample. In this manner a shuffle regressor is obtained. The *for loop* and its arithmetic operations are used to compute the output of the adaptive filter $y[n]$ (equation 3). The subtraction function on the block diagram computes the error signal $e[n]$. This error is multiplied by the step-size μ and then by the elements in the input buffer to obtain the coefficient updates. These updates are added to the previous coefficients vector $\boldsymbol{W}[n-1]$ to compute the updated coefficients vector $\boldsymbol{W}[n]$, described by equation (2). The main advantage of this architecture is that the prescription for the number of coefficients is done by setting a fixed dimension for the vectors $\boldsymbol{W}[n]$ and $\boldsymbol{X}[n]$. Except this setting there is no need for other modifications to obtain a higher order adaptive filter. In this manner the architecture is available for a wide range of filter orders. After this setting a recompilation for the hardware device is required.

IV. EXPERIMENTAL RESULTS

The experiments regarding noise cancelation were carried out with the National Instrument LabVIEW FPGA programming environment, which offers real-time processing of samples.

In this paper the input signals were set as follows: in the first set of experiments a sine wave with *1V* amplitude and *1kHz* frequency (sampled with *100kHz*, *≈17,000* samples)

was used; in a second set of experiments a speech signal (*1.53Vp-p*, *22.05kS/s*, *≈200,000* samples, *1channel/16bits*) was used. These signals were compromised by a white noise with different SNR. In the first set of experiments the input signals were generated with the NI cRIO-9263 analog output module and acquisitioned with the NI cRIO-9215 analog input module. Concerning the second set of experiments these were read from wave files from the real-time controller module. After reading they were sent to a FIFO memory of the FPGA chassis where they were processed by the hardware architecture and finally they were sent back to the real-time controller module to be evaluated. The FIFO memories have the length set to *1023* samples. The processing and reading of the signals is done sample-by-sample from the FIFO's, thus the algorithm works on-line.

The accuracy for the processed signals and for the filter coefficients was set to *20bit*.

Regarding the FPGA synthesis results, a report was generated by the Xilinx FPGA compiler. Its summary is presented in Table 1, for two filter lengths. In order for the project to work on FPGA, the maximum frequency listed in the table must be greater than the actual chassis clock frequency (*40 MHz*). As a measure of the FPGA usage, we can take into consideration the percentage of used slices: *22%* for a filter length of *8*, and respectively *35%* for a filter length of *16*.The FPGA hardware usage also depends on the number of bits describing the process accuracy. Thus, if the hardware usage is the priority then a lower filter length is recommended or the signals accuracy must be decreased.

Filter length	Max. frequency on FPGA (MHz)	Slice Flip Flops out of *28,672*	4 input LUTs out of *28,672*	MULT 18X18s out of *96*	Nr. of slices out of *14,336*
8	*42.784*	*4,110 (14%)*	*5,144 (17%)*	*12 (12%)*	*3,269 (22%)*
16	*42.911*	*6,228 (21%)*	*8,379 (29%)*	*12 (12%)*	*5130 (35%)*

Table 1. LabVIEW FPGA hardware usage.

In order to illustrate the capability of the algorithm we chose two cases regarding the input signal. The results are summarized in Table 2; the step-size was set to *1E-4*. Figures 4, 5 and 6 show the tracking ability and a spectrum analysis of the process for different input SNRs. The quality of the process was measured in terms of the filtered signal's SNR and the convergence measured in seconds and samples. The SNR for the output signal is evaluated as:

$$SNR = \frac{RMS(x)}{RMS(y-x)}, \tag{4}$$

where *RMS(x), RMS(y-x)* are the root mean squares of the pure input signal throughout the adaptive filter and the noise at the output. The output noise is estimated as a difference between the filtered and the pure input signals (*y-x*).

Signal type	Filter length	SNR [*dB*]		Convergence	
		Input	Output	[*s*]	samples
Noisy sine	*8*	*7.8*	*17.4*	*0.07*	*6,500*
		15.6	*24.5*	*0.06*	*5,000*
	16	*7.8*	*18.8 (Fig.4a)*	*0.05*	*4,500*
		15.6	*24.6 (Fig.4b)*	*0.04*	*3,500*
Noisy speech (*Fig. 5a, 6a*)	*8*	*6.2*	*13.1*	*5*	*110,000*
		12.2	*17.1 (Fig.5b, 6b)*	*4.1*	*92,000*
	16	*6.2*	*15.1*	*3.7*	*82,000*
		12.2	*18.2 (Fig.5c, 6c)*	*3.3*	*67,500*

Table 2. Quality estimation of the process.

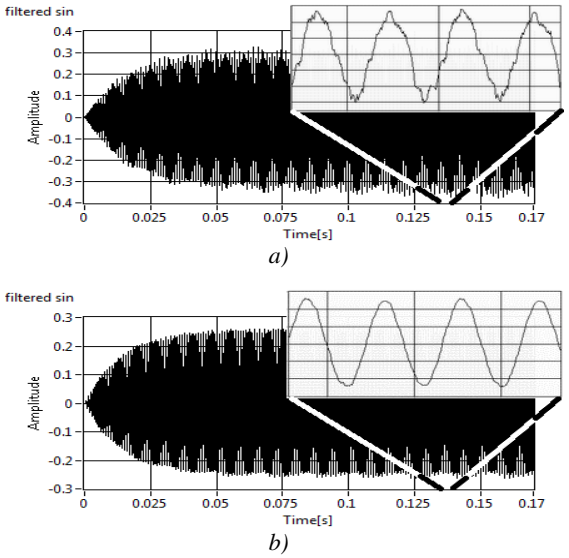

Figure 4. Tracking ability on a sine wave with 16 coefficients LMS filter: a) SNR=7.8dB, b) SNR=15.6dB.

Figure 4 shows the result of the filtering process when the input is a sine wave with different SNRs (*7.8dB* and *15.6dB*); the filter length was set to *16*. As it can be seen the SNR at the output of the adaptive filter was increased considerably in both cases (Table 2).

Figure 5 and 6 illustrate a case in the filtering process of a speech signal for two filter lengths (*8* and *16*) with a *12.2dB* SNR at the input. In Figure 5 the tracking ability of the algorithm is presented. Figure 6 shows the amplitude spectra for the processed waveform shown in Figure 5. The SNR was increased well when the filter length is *16* in comparison with the case when the filter length is *8*. As for the case with SNR of *6.2dB*, this was increased at the output, but not so well than in case from Figure 5 even if the filter length was set to *16* (Table 2).

The same set of analysis with different values for the step-size μ was also performed. In this case the filtered signal's SNR was increased well for $1E\text{-}6 \leq \mu \leq 1E\text{-}4$.

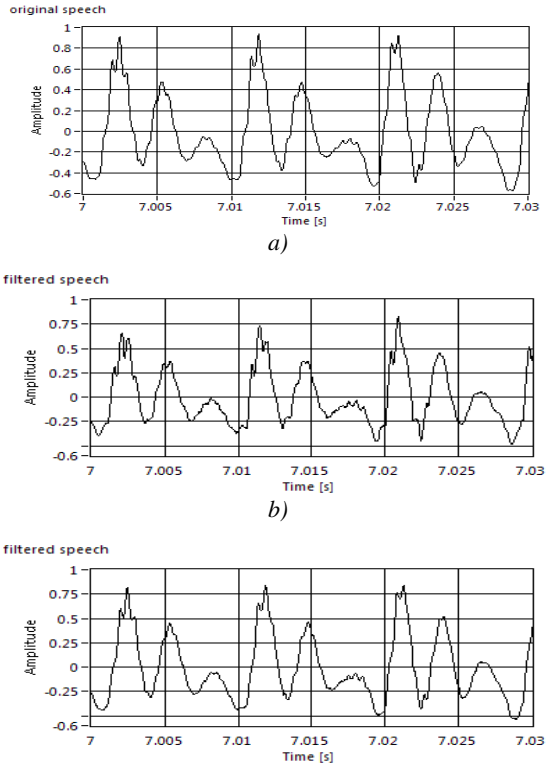

Figure 5. Tracking ability on a speech signal: a) original speech; b) and c) - filtered waveforms with 8 and 16 coefficients LMS filter

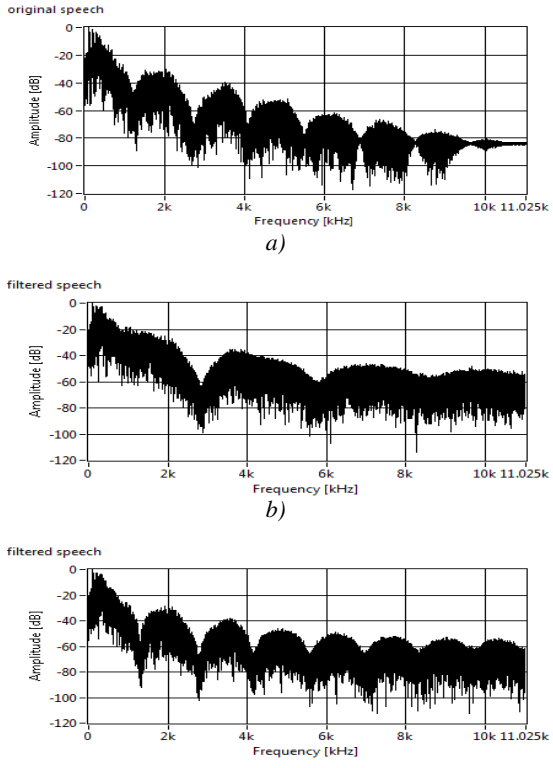

Figure 6. Spectrum analysis on a speech signal: a) original spectrum; b) and c) - filtered spectra with 8 and 16 coefficients LMS filter

V. CONCLUSIONS

This paper presents a hardware architecture used to implement the LMS adaptive algorithm for several filter lengths. The architecture is aimed for speech processing using the NI cRIO-9104 FPGA chassis and the LabVIEW FPGA toolkit.

The performance of the LMS algorithm implemented by hardware is comprehensively analyzed in terms of convergence performance, filtered signal's SNR, filter length and tracking ability. The experimental results ensure the feasibility of the high-speed FPGA architecture of the LMS algorithm (Table 1). The noise cancelling system is chosen to validate the performance of LabVIEW FPGA in signal processing applications (Table 2); FPGA implementation will be more effective for more complex digital signal processing systems.

Various DSP applications that use LMS adaptive FIR filters, like echo cancellers, channel equalizers and noise cancellers, can be added as a software layer on top of our system, without any hardware modifications.

The presented architecture may be further enhanced so that it has the ability to initialize the filter coefficients. This feature may be needed for applications requiring faster convergence.

VI. ACKNOWLEDGEMENTS

The authors deeply thank National Instruments for providing support of LabVIEW FPGA hardware/software tools and for their generous guidance, support and training.

REFERENCES

[1] A. Elhossini, S. Areibi, R. Dony, "An FPGA Implementation of the LMS Adaptive Filter for Audio Processing", *IEEE International Conference on Reconfigurable Computing and FPGA's*, 2006

[2] S. Prakash, D.V. Venkatasubramanyam, B. Krishnan, R. Nagendra, "Compact FPGA Controller Aircraft/Aerospace Structures", *Proceedings of the International Conference on Aerospace Science and Technology*, India, 2008

[3] E. C. Ifeachor and B. W. Jervis, *Digital Signal Procesing, A Practical Approach*, Prentice Hall, 2002

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

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

[6] E. Szopos, M. Topa, I. Dornean, R. Groza, "Adaptive LMS Algorithm System Identification using LabVIEW", *IEEE International Conference on Automation, Quality and Testing, Robotics*, Romania, pp.254-257, 2008.

[7] E. Szopos, M. Topa, R. Groza, "Adaptive LMS and Pade Algorithms in System Identification", *Applied Electronics International Conference Pilsen*, pp.219-222, 2008

[8] Sinead Mullins, Conor Heneghan, "Alternative Least Mean Square Adaptive Filter Architectures for Implementation on FPGA**",** *Digital Signal Processing Group, Department of Electronic and Electrical Engineering, University College Dublin.*

[9] www.ni.com

[10] T. Lan, J. Zhang, "FPGA Implementation of an Adaptive Noise Canceller", *IEEE International Symposiums on Information Processing*, 2008