Volume 50, Number 4, 2009

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

COMPARISON OF LMS ALGORITHM DERIVATIVES USING LABVIEW FPGA

Erwin SZOPOS[1] Horia HEDESIU[2] Victor POPESCU[1] Lelia FESTILA[1]
[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, Victor.Popescu@bel.utcluj.ro, Lelia.Festila@bel.utcluj.ro*

Abstract: The LMS algorithm derivatives have found wide application in many areas of adaptive signal processing and control. In this paper we study the tracking performance of the LMS derivatives in adaptive noise cancellation for a speech signal corrupted by white noise. The experiments were done for 16–tap filter lengths and as for the hardware platform the NI cRIO-9104 FPGA chassis and the NI cRIO-9014 real-time module were used. The obtained results were compared to highlight the performances of each of the LMS derivatives. The results of the experiments confirm that the NLMS variant provides the most enhanced SNR and the better tracking ability is offered by the LLMS in a 16-tap adaptive filter. The performances of the algorithms are evaluated in terms of signal-to-noise ratio (SNR) at the output of the adaptive system, tracking ability and hardware usage.

Keywords: *LMS derivatives, FPGA, SNR, pipeline, tracking, hardware usage.*

I. INTRODUCTION

One of the most popular adaptive algorithms available in the literature is the Least Mean Square (LMS) algorithm [1]. When a linear combiner and an adaptive algorithm are joined together a linear adaptive filter is obtained. The main property of an adaptive filter is he tracking capability. Tracking occurs when the adaptive filter is used in a non-stationary environment, for which the optimum Wiener solution takes on a time-varying form. In such a situation the adaptive filter is expected to not only adapt the filter weights nearly of their optimum values, but also to follow the variations of the optimum tap weights.

The input signals are stochastic, and information obtained from the inputs is used by the adaptive algorithm to adjust the weights. An efficient algorithm minimizes the usage of data while maximizing the quality of the solution, achieving parameter adjustments close to optimum. Minimizing data usage corresponds to fast adaptive convergence, but fast convergence could provide a poor quality of solution.

The aim of this paper is to present a comparison of the LMS derivatives behavior in same conditions using the LabVIEW FPGA toolkit. These algorithms are used to filter a speech signal to enhance its SNR. The algorithms are implemented on the National Instruments cRIO-9104 FPGA chassis [2]. The performances of the architectures are evaluated in terms of SNR measured at the output of the adaptive filter, tracking ability and FPGA resource usage.

This paper is organized as follows: in section II the theoretical overview regarding the LMS algorithm and its derivatives, the processing modules and tools used for implementation are given; in section III the implementation of the architectures are presented; section IV presents the experimental results, and finally in section V the main conclusions of the paper are drawn.

II. THEORETICAL OVERVIEW

A. THE LMS ALGORITHM AND ITS DERIVATIVES

The LMS algorithm is a gradient descent algorithm as it uses the gradient vector of the filter weights to converge on the optimal Wiener solution. It is a tradeoff between computational simplicity and performance. The weight vector update of the conventional LMS algorithm is [3]:

$$W[k] = W[k-1] + \mu \cdot e[k] \cdot X[k-1], \tag{1}$$

where k is the time step, μ is the *step-size* that controls the stability and the convergence rate: $0 < \mu < 2/\lambda_{\max}$ [1].

The NLMS algorithm is a normalized variant of the conventional LMS; per iteration it uses a step-size that is computed in terms of the input data norm. Thus, the normalized μ is adjusted with the input data as:

$$\mu[k] = \frac{\mu_0}{\alpha_0 + \|X[k]\|}, \tag{2}$$

where $0 < \mu_0 < 2$, α_0 is a small constant to avoid division by zero and $\|X[k]\| = \sqrt{\sum x^2[k]}$. The NLMS algorithm has a convergence speed that is potentially higher than the conventional LMS, for uncorrelated as well as correlated

Manuscript received June 26, 2009; revised November 17, 2009

input signals. Thus, the filter weight vector update for this algorithm is as follows [3]:

$$W[k] = W[k\text{ - }1] + \frac{\mu_0}{\alpha_0 + \|X[k]\|} \cdot e[k] \cdot X[k\text{ - }1]\,. \tag{3}$$

Three similar LMS derivatives are the ones that use a *sign* function about a quantity to compute the weight vector instead of magnitude of the same quantity. These are the *sign-error* LMS (SELMS), the *sign-data* LMS (SDLMS) and the *sign-sign* LMS (SSLMS) algorithms [3]. These derivatives are useful in practice due to the simplicity of digital hardware implementation, as the error value need not be exact or stored during adaptation. The *sign* LMS derivatives are used in applications where very high speed data is being processed, such as communications equipment. The weight vector updates for these derivatives are:

$$\begin{aligned} &SELMS: \; W[k] = W[k\text{ - }1] + \mu \cdot sign\{e[k]\} \cdot X[k\text{ - }1] \\ &SDLMS: \; W[k] = W[k\text{ - }1] + \mu \cdot e[k] \cdot sign\{X[k\text{ - }1]\} \\ &SSLMS: \; W[k] = W[k\text{ - }1] + \mu \cdot sign\{e[k]\} \cdot sign\{X[k\text{ - }1]\} \end{aligned} \tag{4}$$

Between the *sign* LMS algorithms the fastest is the SDLMS one, but it is not faster than the conventional LMS.

In cases where the Wiener problem is ill-conditioned, and multiple solutions exist the *leaky* LMS algorithm (LLMS) is used [3], [4]:

$$W[k] = (1 - \alpha\mu) \cdot W[k\text{ - }1] + \mu \cdot e[k] \cdot X[k\text{ - }1]\,, \tag{5}$$

where $0 < \alpha \ll 1$. This algorithm not modifies substantially the original Wiener–Hopf solution but it accelerates the convergence of the LMS algorithm.

Another popular variant of LMS is the *momentum* LMS (MLMS) that uses the weight vector update equation [5]:

$$W[k] = W[k\text{ - }1] + \alpha(W[k\text{ - }1]\text{ - }W[k\text{ - }2]) + \mu \cdot e[k] \cdot X[k\text{ - }1]\,. \tag{6}$$

The momentum constant (α) controls the scale of the gradient of previous filter weight [6]. If the previous gradient is relative large then this additional gradient term will accelerate the search towards the global minimum.

There are obviously many variants of these types of algorithms involving heuristics and higher-order approximations to the derivative. Each of the LMS derivatives adjusts the weights of the filter so that the mean square error is minimized:

$$MSE = \frac{1}{N}\sum_{n=0}^{N-1} e^2[k] = \frac{1}{N}\sum_{n=0}^{N-1} [d[k] - y[k]]^2 \tag{7}$$

where $e[k]$ and $d[k]$ are the error and the desired output signals. The output vector of the adaptive filter is as follows:

$$Y[k] = W[k\text{ - }1]^T \cdot X[k] \tag{8}$$

Figure 1 shows the block diagrams of the weight vector update for some of the LMS derivatives.

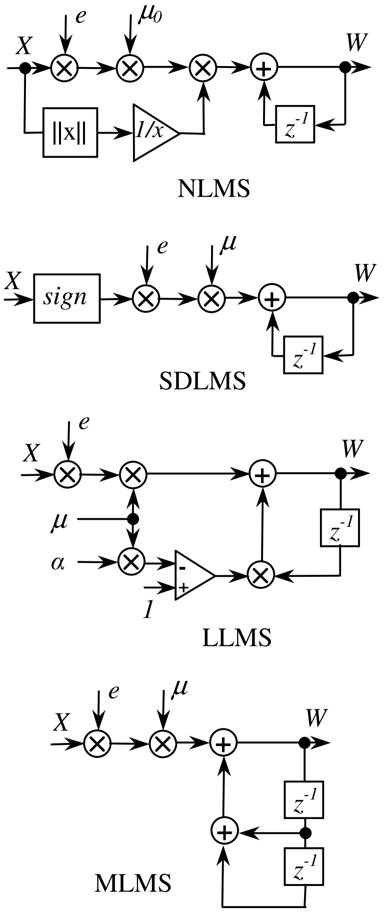

Figure 1. Weight vector computation for LMS derivatives

B. THE USED HARDWARE

The National Instruments cRIO-9104 reconfigurable embedded chassis [2] is used to implement the adaptive hardware architectures. 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 controller 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 LabVIEW suite is used to implement the hardware architectures for signal processing.

III. FPGA IMPLEMENTATION

A LabVIEW FPGA implementation of the conventional LMS adaptive algorithm was developed and presented in [7]. To implement the derivatives, the architectures was first coded in LabVIEW FPGA. Simultaneously, a thorough testbench was written in LabVIEW software to verify each of the adaptive filter core. The testbench provides the input test vectors to the core simulating the actual processor-based system. To analyze and compare the behavior of the LMS derivatives (eq. (3), (4), (5), (6)) the application of noise cancellation was selected [8], [9]. The algorithms were tested simultaneous to compare their behavior in same conditions.

Figure 2 shows the LabVIEW FPGA architectures for two of the LMS derivatives based on the architecture presented in [7]. These architectures were optimized for speed in comparison to [7], using the pipeline technique. Pipelining is used to parallelize an inherently serial section of code in the applications. Applications are good candidates for pipelining if they have several stages, or sections, of code that execute in sequence. The goal of pipelining is to separate out these stages to run in assembly-line fashion. In this paper pipelining is implemented in LabVIEW with the usage of so called *shift registers* [2]. Another way to parallelize the code is the usage of *feedback node*. The initialization of the shift registers and the feedback nodes is done externally with a value of zero. The pipeline technique with feedback node uses less hardware and can process with higher speed than with shift registers.

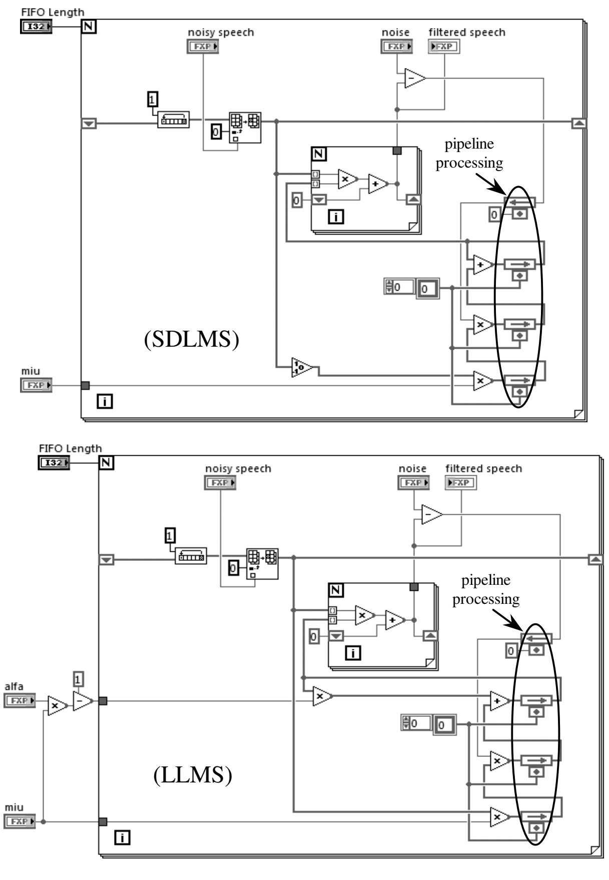

Figure 2. LabVIEW FPGA architectures for SDLMS and LLMS algorithms using pipeline with feedback node

For these architectures the number of weights can be increased or decreased as well, but recompilation is required.

IV. EXPERIMENTAL RESULTS

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

In this paper the experiments were done using speech signals, the inputs were compromised by white noise (different SNRs); the step-size and the filter's length were set to *1E-4* and *16*.

The objective of this work is to compare the LMS derivatives using LabVIEW FPGA hardware description in signal filtering [8], [10]. The aim of these algorithms is to enhance the SNR of an input speech signal. The quality of the process was evaluated in terms of the filtered signal's SNR, tracking ability and hardware usage.

To illustrate the performances of the algorithms a speech signal with the length of *9 sec* and with *12.2dB* SNR was chosen. Table 1 summarizes the efficiency of each algorithm proving that the NLMS algorithm provides the most enhanced SNR at the output. In respect of the SNR measured at the output this algorithm is followed by the MLMS one. Concerning the hardware usage (Table 1) the LMS algorithm uses less hardware area and also may work at higher speed. The MLMS algorithm has good tracking ability but it uses more hardware area. Thus, if the hardware usage is the priority then the LMS algorithm is recommended; if the smoothing is the priority then the NLMS algorithm is preferred.

Alg. Type	SNR [*dB*]		Hardware usage	
	Input	Output	Slice Flip Flops out of *28,672*	Max. frequency on FPGA (MHz)
LMS	*12.2*	*19.05*	*25%*	*42.015*
NLMS		*23.92* $\mu_0=0.0001$ $\alpha=0.001$	*27%*	*40.518*
SDLMS		*17.67*	*26%*	*40.811*
LLMS		*19.07* $\alpha=0.1$	*28%*	*40.449*
MLMS		*20.47* $\alpha=0.5$	*37%*	*40.621*

Table 1. Quality estimation of the LMS derivatives

Figure 3 shows the input noisy speech in comparison with the smooth one. Figures 4 show the tracking ability of the LMS derivatives included in Table 1. One can see that each of the algorithms have good tracking ability but the best is the LLMS one. On the other hand a delay between the input and output signals can be observed, which is introduced by the adaptive algorithms. No significant delay was obtained in the steady-state regime when the pipeline technique was used.

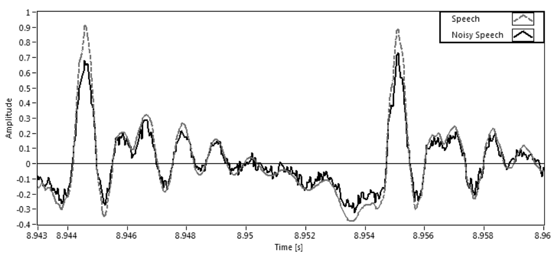

Figure 3. The input noisy speech vs. the original one

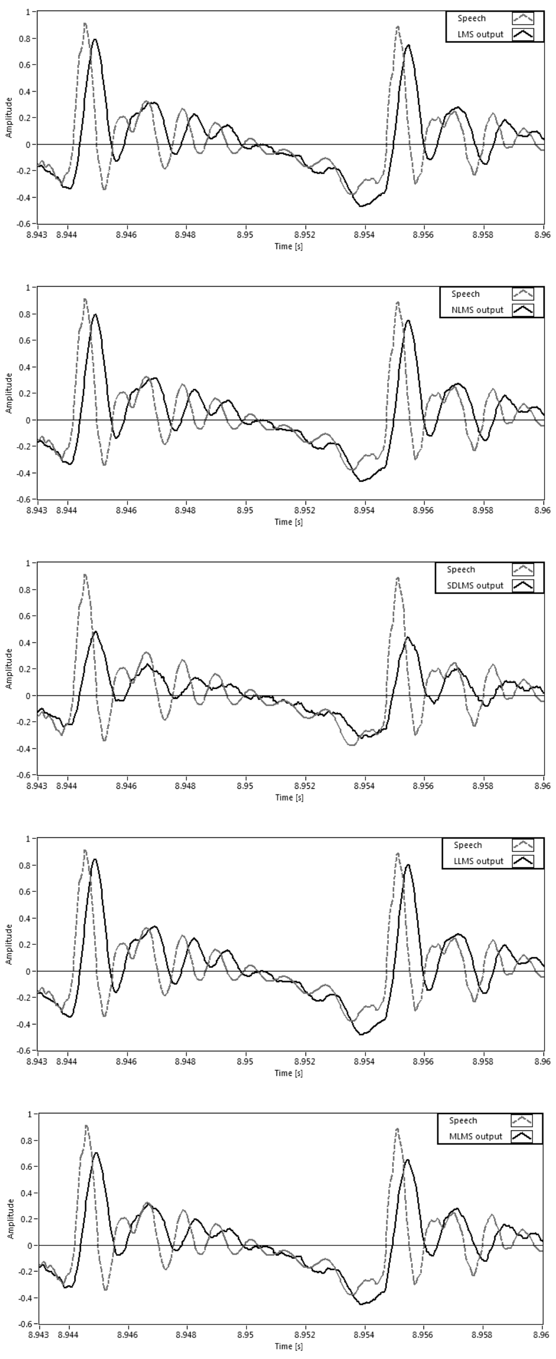

Figure 4. Tracking ability of the LMS derivatives (LMS, NLMS, SDLMS, LLMS and MLMS) on a speech signal

V. CONCLUSIONS

To obtain the optimum weight vector in the LMS-type algorithms a difference between the desired output and the estimated one has to be minimized for all the input vectors. On the implementation of the LMS algorithm and its derivatives, one can say that they are very simple and easy to apply due to the simple arithmetical computations. The algorithms provide convergence and stability although there is a noise in the input signal.

In this paper we compared the LMS derivatives implemented on the NI cRIO-9104 FPGA chassis and the NI cRIO-9014 real-time module. The implementations of the algorithms were carried out in fixed-point because this representation is more understandable and is easy to verify with diverse simulation programs.

The comparison process concerns the tracking performances and signals smoothing in the steady-state regime. The hardware usage for each of the algorithms was also analyzed and presented. The experimental results prove the LLMS best tracking ability, the NLMS best smoothness and the LMS less hardware usage. The presented architectures were speed optimized by the usage of the pipeline technique.

Researches continue respecting the algorithms architecture for convergence speed enhancement.

VI. ACKNOWLEDGEMENTS

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

REFERENCES

[1] B. Widrow, M.A. Lehr, “30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation”, *Proceedings of the IEEE*, Vol. 78, No. 9, 1990

[2] www.ni.com

[3] D. Mali, “Comparison of DC Offset Effects on LMS Algorithm and its Derivatives”, *International Journal of Recent Trends in Engineering*, Vol 1, No. 1, 2009

[4] M. Kamenetsky, B. Widrow, “A Variable Leaky LMS Adaptive Algorithm," *Proceeding Asilomar Conference on Signals, Systems and Computers,* USA, 2004

[5] L.K. Ting, C.F.N. Cowan, R.F. Woods, “Tracking Performances of Momentum LMS Algorithm for a Chirped Sinusoidal Signal”, *Proceedings of X European Signal Processing Conference*, Finland, 2000

[6] R. Scharma, W.A. Sethares, J.A. Bucklew, “Analysis of Momentum Adaptive Filtering Algorithms,” *IEEE Transactions on Signal Processing*, Vol. 46, No. 5, 1998

[7] E. Szopos, H. Hedesiu, “LabVIEW FPGA Based Noise Cancelling Using the LMS Adaptive Algorithm”, *Acta Technica Napocensis*, Vol. 50, No. 4, 2009

[8] W. Fohl, J. Matthies, B. Schwartz, “A FPGA-Based Adaptive Noise Cancelling System”, *Proceeding of the 12th International Conference on Digital Audio Effects*, Italy, 2009

[9] B. Widrow, J.R. Glover, J.M. McCool and others, “Adaptive Noise Cancelling: Principles and Applications” *Proceedings of the IEEE*, Vol. 64, No. 12, 1975

[10] S.S. Godbole, P.M. Palsodkar, V.P. Raut, “FPGA Implementation of Adaptive LMS Filter”, *Proceedings of SPIT-IEEE Colloquium and International Conference*, India, 2008