FPGA IMPLEMENTATION OF A VARIABLE STEP-SIZE AFFINE PROJECTION ALGORITHM FOR ACOUSTIC ECHO CANCELLATION

Cristian Anghel[1], *Constantin Paleologu*[1], *Jacob Benesty*[2], *and Silviu Ciochină*[1]

[1] Department of Telecommunications, University Politehnica of Bucharest
1-3, Iuliu Maniu Blvd., 061071, Bucharest, Romania
email: {canghel, pale, silviu}@comm.pub.ro
[2] INRS-EMT, Universite du Quebec
QC H5A 1K6, Montreal, Canada
email: benesty@emt.inrs.ca

ABSTRACT

This paper presents a field-programmable gate array (FPGA) implementation of a recently proposed variable step-size affine projection algorithm (VSS-APA), in the context of acoustic echo cancellation. The proposed hardware implementation scheme takes advantage of the algorithm's specific features. Area and speed results are provided for the Xilinx Virtex 5 XC5VFX70T chip from the Xilinx ML507 evaluation board, when considering the particular case of the projection order p = 2. The overall performance of this acoustic echo canceller (AEC) indicates that it could be a reliable solution for real-world acoustic echo cancellation scenarios.

1. INTRODUCTION

In acoustic echo cancellation scenarios, an adaptive filter identifies the acoustic echo path between the terminal's loudspeaker and microphone, i.e., the room impulse response [1]. The affine projection algorithm (APA) [2] and different versions of it are reliable candidates for acoustic echo cancellation. As compared to the normalized least-mean-square (NLMS) algorithm, the APA achieves faster convergence rate and tracking, with only a moderate increase of computational complexity. In a similar way to the NLMS, the performance of the APA is controlled by the value of its step-size. This parameter compromises between the convergence rate and misadjustment. In order to optimize this trade-off, several variable step-size APAs (VSS-APAs) were developed (e.g., see [3] and the references therein).

The VSS-APA proposed in [3] was designed in the context of acoustic echo cancellation. The main advantages of this algorithm are its non-parametric nature (i.e., it does not require any *a priori* information about the acoustic environment), together with robustness against near-end signal variation (e.g., double-talk). Thanks to these features, this algorithm could be a reliable candidate for real-world applications. However, the capabilities of an algorithm could be seriously affected when using practical implementation platforms, e.g., a field-programmable gate array (FPGA). In this context, several finite-precision effects could significantly bias the acoustic echo canceller (AEC) behavior.

There are two antagonistic aspects that need to be considered when dealing with a hardware implementation of an AEC on FPGA devices using very high speed integrated circuit hardware description language (VHDL). The first is the relation between the values of the sampling frequency and the system clock frequency, which allows a large number of operations between two successive samples. The second is the sensitivity of the adaptive algorithm to quantization errors, which makes the number of bits used for representation have a great impact on the AEC performance. Therefore, the implementation may benefit by the module reuse-in-time advantage that tends to reduce the occupied area; but it also may require a large number of bits for representation (in order to get closer to infinite precision simulation results), which further increases the number of used resources.

In the literature, there are many examples of hardware implementation of AEC, based on different kinds of adaptive algorithms. The authors themselves proposed some of them, most recently in [4]. In this paper, we present an AEC implemented on a Xilinx Virtex 5 FPGA (i.e., XC5VFX70T chip) [5], [6], based on the VSS-APA proposed in [3]. This algorithm is briefly presented in Section 2. The proposed hardware implementation scheme of the AEC is detailed in Section 3 for the particular case of the projection order $p = 2$. Simulation results are given in Section 4. Finally, Section 5 concludes this work.

2. VSS-APA FOR ACOUSTIC ECHO CANCELLATION

In the following, all signals are real-valued and the time index is denoted by n. The main goal of an AEC is to identify an unknown system (i.e., acoustic echo path) using an adaptive filter. In this context, the far-end signal, $x(n)$, coming from the loudspeaker goes through the room impulse response, providing the echo signal. This signal is added with the near-end signal, resulting the microphone signal, $y(n)$. The adaptive filter of length L, defined by the vector $\hat{\mathbf{h}}(n) = [\hat{h}_0(n), \hat{h}_1(n), \ldots, \hat{h}_{L-1}(n)]^T$ (where superscript T denotes transposition), aims to produce at its output an estimate of the echo, $\hat{y}(n)$, while the error signal, $e(n) = y(n) - \hat{y}(n)$, should contain an estimate of the near-end signal.

The VSS-APA proposed in [3] is defined by the following relations:

$$\mathbf{e}(n) = \mathbf{y}(n) - \mathbf{X}^T(n)\hat{\mathbf{h}}(n-1), \qquad (1)$$

 ISSN 2076-1465

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mathbf{X}(n)\mathbf{N}^{-1}(n)\boldsymbol{\mu}(n)\mathbf{e}(n), \quad (2)$$

where $\mathbf{y}(n) = [y(n), y(n-1), \ldots, y(n-p+1)]^T$ is the reference signal vector of length p (with p denoting the projection order), $\mathbf{N}(n) = \mathbf{X}(n)^T\mathbf{X}(n) + \delta\mathbf{I}_p$, δ is a regularization parameter, and $\mathbf{I}_p$ is the pxp identity matrix. The matrix $\mathbf{X}(n) = [\mathbf{x}(n), \mathbf{x}(n-1), \ldots, \mathbf{x}(n-p+1)]$ is the input signal matrix, where $\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-L+1)]^T$, and $\boldsymbol{\mu}(n)$ is the step-size matrix defined as

$$\boldsymbol{\mu}(n) = \mathrm{diag}\left\{\mu_0(n), \mu_1(n), \ldots, \mu_{p-1}(n)\right\}. \quad (3)$$

These elements are evaluated for $l = 0, 1, \ldots, p-1$ as follows:

$$\mu_l(n) = \left| 1 - \frac{\sqrt{\left|\hat{\sigma}_y^2(n-l) - \hat{\sigma}_{\hat{y}}^2(n-l)\right|}}{\xi + \hat{\sigma}_{\mathbf{e}_{l+1}}(n)} \right|. \quad (4)$$

A small positive number ξ avoids division by zero and $\mathbf{e}_{l+1}(n)$ denotes the $(l+1)$-th element of the vector $\mathbf{e}(n)$; in a general manner, a parameter of the form $\hat{\sigma}_\alpha^2(n)$ denotes the power estimate of the sequence $\alpha(n)$, and can be computed as

$$\hat{\sigma}_\alpha^2(n) = \lambda\hat{\sigma}_\alpha^2(n-1) + (1-\lambda)\alpha^2(n), \quad (5)$$

where λ is a weighting factor [7]. Since only the parameters available from the adaptive filter are required in (4) and there is no need for *a priori* information about the acoustic environment, this VSS-APA is easy to control in practice. The simulation results presented in [3] indicate that a value of the projection order $p = 2$ offers a good compromise between the convergence rate, misadjustment, and complexity. Therefore, the following FPGA implementation considers only this case.

3. PROPOSED HARDWARE IMPLEMENTATION SCHEME

The algorithm described in Section 2 contains some challenging operations to implement, i.e., the fractional division, square-root, and matrix inversion. Also, modules like fractional multipliers with different input widths and two input summing units have to be implemented. Besides these processing modules, some pre-processing modules are required due to the very small numerical values obtained for some of the algorithm's parameters (e.g., power estimates of the error signal, the value of the matrix's determinant). These modules will be described in the following sub-sections. Their design is adapted to the application's particularities, in order to provide modularity to the proposed implementation scheme. Finally, a block scheme of the AEC will be presented in the last sub-section. All the area and speed results will refer to Xilinx Virtex 5 XC5VFX70T chip [5] from the Xilinx ML507 evaluation board [6]. However, the area result is provided in terms of 6-input look-up tables (LUTs), flip-flops (FFs) and block RAMs (BRAMs), so it should characterize any FPGA with the same architecture. The percentage of the used area shows that a much smaller FPGA could be used for this implementation.

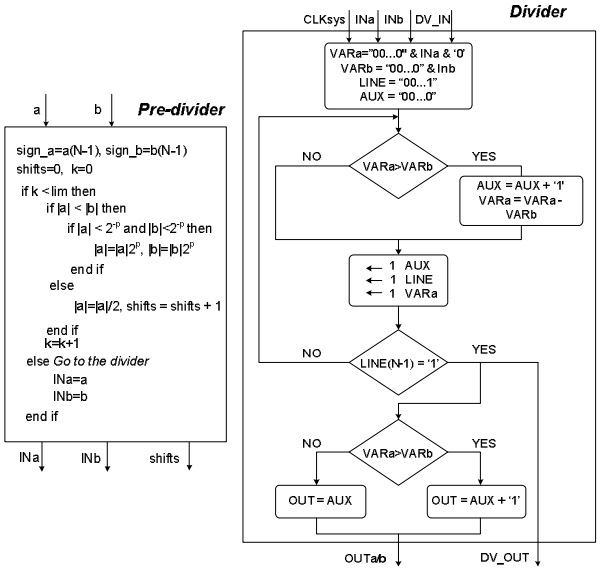

Figure 1 – Pre-divider procedure and divider block scheme.

3.1 Fractional multiplier

The fractional multiplier module has a large reuse factor in the AEC scheme. When implementing the VSS-APA with $p = 2$, two multipliers are needed for computing the product $\mathbf{X}^T(n)\hat{\mathbf{h}}(n-1)$; other multipliers are required for computing the matrix product $\mathbf{N}^{-1}(n)\boldsymbol{\mu}(n)\mathbf{e}(n)$. The system clock frequency is approximately 200 MHz and the sampling frequency is 8 kHz; consequently, there are approximately 25000 clock periods available between two successive samples. This observation allows us to increase the reuse factor for each module. Also, it indicates that no additional stress should be placed on the critical path (from the point of view of the maximum delay) if this frequency is reached.

There is a time relation between the above mentioned operations. First, the product $\mathbf{X}^T(n)\hat{\mathbf{h}}(n-1)$ needs to be evaluated in order to compute the error vector $\mathbf{e}(n)$; then, having this error vector, the power estimates of its components are evaluated in order to be used for computing the step-sizes, i.e., the elements of $\boldsymbol{\mu}(n)$. At the same time, the operations required for obtaining the matrix $\mathbf{N}^{-1}(n)$ can be performed. The proposed implementation scheme contains two binary tree multipliers for the first set of operations. Their inputs are time-multiplexed for all the required operations and the results are obtained in a pipe-line manner. However, the price to pay for this feature is an increase of the occupied area. For the second set of operations a single sequential multiplier is used. Its inputs are also time-multiplexed, but the results are obtained once at N_b clock period, where N_b is the number of bits of the operand with the larger width. The main advantage is the reduced occupied area.

3.2 Fractional divider

This module is used in the square-root approximation algorithm, in updating the step-size values, and in the inverse matrix computation.

```
function sqrt_a=sqrt(a)
    x=1; k=0;
    y=(x+a/x)/2
    if k<Niter then
       if (|x-y|>2^-r) then
          x=y; y=(x+a/x)/2;
       else
          x=x;
       end
       k=k+1;
    end
    sqrt_a=x
```

Figure 2 – Square-root approximation algorithm.

For the fractional division operation, the dividend has to be less than or equal to the divisor. In order to accomplish this requirement, the divider module is preceded by a pre-divider unit that compares the two inputs. The testing unit right shifts the dividend until it becomes less then or equal to the divisor. The number of shifted positions is stored and compensated by left shifting the terms of the inverse matrix that multiplies the quotient of the divider. The pre-divider unit also scales with the same amount (when possible) both the dividend and the divisor in order to increase their values for more precise results at the output of the dividers (Fig. 1). This remark is true because the width of variables is higher than that used in the divider module.

These processes are applied a fixed number of times, in order to obtain the same latency. For two N-bit width inputs, the quotient is obtained after N clock periods from the divider unit. A delay line of length N is used to count this time period. The logic diagram of the unsigned divider is given in Fig. 1. The signs of the two operands are used after the dividing operation, in order to produce the correct result. As one can observe, the divider module uses only shifts, tests, and adders, so it is very suitable for hardware implementation.

3.3 Square-root module

The square-root is the most challenging operation needed in the step-size computation. The approximation algorithm is described in Fig. 2; it is based on the property of the sequence $c_n = (c_{n-1} + a/c_{n-1})/2$ which converges to $a^{1/2}$. The simulations show that a number *Niter* = 12 iterations produces a very good approximation of the square-root function. When the result is ready before *Niter*, the algorithm ends and the result is buffered in order to produce the same processing delay. We can efficiently use the number *Niter* by choosing a proper value of the threshold 2^{-r} (see Fig. 2).

3.4 Inverse matrix computation

For the inverse matrix computation, we propose an optimized scheme due to the specific operand structure. First, the product $\mathbf{M}(n) = \mathbf{X}^T(n)\mathbf{X}(n)$ needs to be computed. For $p = 2$, the matrix $\mathbf{M}(n)$ becomes

$$\mathbf{M}(n) = \begin{bmatrix} \sum_{k=n-L+1}^{n} x^2(k) & \sum_{k=n-L+1}^{n} x(k)x(k-1) \\ \sum_{k=n-L+1}^{n} x(k)x(k-1) & \sum_{k=n-L}^{n-1} x^2(k) \end{bmatrix}$$

and its elements can be computed recursively as follows:

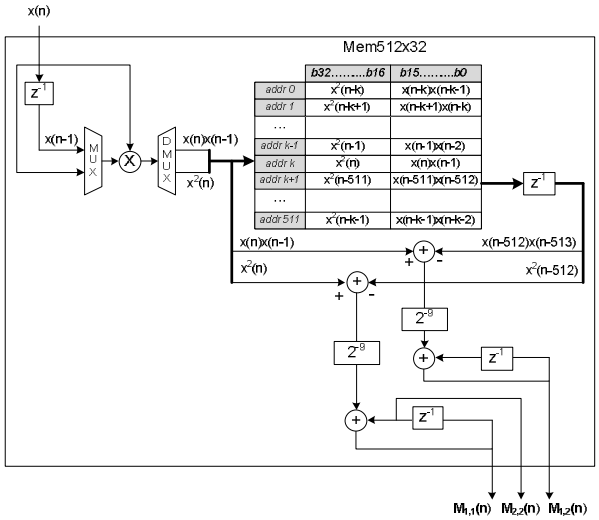

Figure 3 – Matrix inversion diagram.

$$\mathbf{M}_{1,1}(n) = \mathbf{M}_{1,1}(n-1) + x^2(n) - x^2(n-L),$$
$$\mathbf{M}_{1,2}(n) = \mathbf{M}_{1,2}(n-1) + x(n)x(n-1) - x(n-L)x(n-L-1),$$
$$\mathbf{M}_{2,1}(n) = \mathbf{M}_{1,2}(n), \quad \text{and} \quad \mathbf{M}_{2,2}(n) = \mathbf{M}_{1,1}(n-1).$$

The proposed scheme implements these recursive relations by using a single block memory and a single sequential multiplier, as in Fig. 3 (considering a filter length L = 512). The memory acts like a circular buffer, i.e., writing at address k and reading from address $k + 1$. The address is incremented modulo L. Then, the new matrix $\mathbf{N}(n) = \mathbf{M}(n) + \delta\mathbf{I}_p$ and its determinant need to be computed, providing the outputs of this block. The dividers are not included inside this module. Although the recursive manner of computation could be sensitive to quantization errors when using a finite precision representation, the trade-off between the used implementation resources and the performances degradation is an acceptable one, as it will be described in Sections 3.8 and 4.

3.5 Step-size computation

The computation of the elements of $\boldsymbol{\mu}(n)$ is based on (4). This module contains a divider (with its corresponding pre-divider) and a square-root module. These modules are used time-multiplexed for the elements of $\boldsymbol{\mu}(n)$. These operations are made after the computation of the error signal vector.

3.6 Filter coefficient update

The filter coefficients are updated after computing some intermediate terms. For making the first step in computing the inverse of the matrix $\mathbf{N}(n)$, the product

$$\begin{bmatrix} \mathbf{T}_{1,1}(n) & \mathbf{T}_{1,2}(n) \\ \mathbf{T}_{2,1}(n) & \mathbf{T}_{2,2}(n) \end{bmatrix} = \begin{bmatrix} \mathbf{N}_{2,2}(n) & -\mathbf{N}_{2,1}(n) \\ -\mathbf{N}_{1,2}(n) & \mathbf{N}_{1,1}(n) \end{bmatrix} \begin{bmatrix} \mu_0(n) & 0 \\ 0 & \mu_1(n) \end{bmatrix}$$

needs to be evaluated using two binary tree multipliers for two successive input pairs, i.e., $\mathbf{N}_{2,2}(n)\mu_0(n)$ and $-\mathbf{N}_{1,2}(n)\mu_0(n)$, respectively $-\mathbf{N}_{2,1}(n)\mu_1(n)$ and $\mathbf{N}_{1,1}(n)\mu_1(n)$.

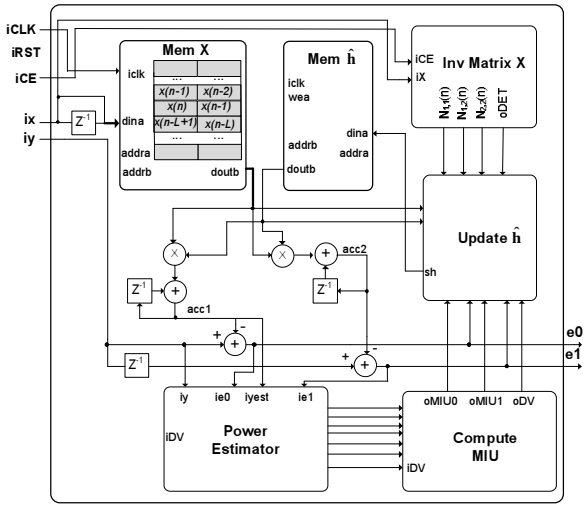

Figure 4 – AEC block scheme.

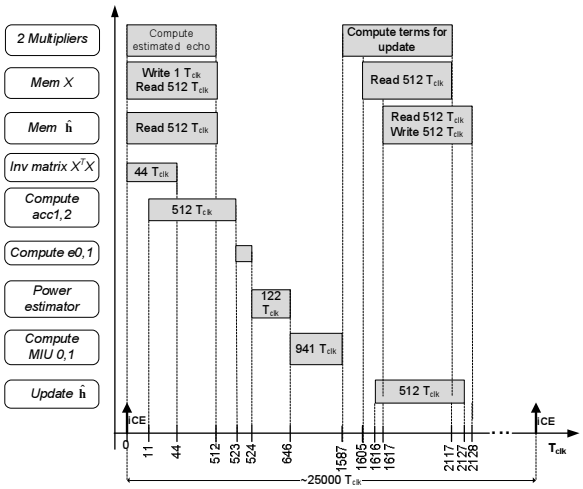

Figure 5 – Timing diagram of the AEC.

It should be noticed that the elements of the matrix $\mathbf{N}(n)$ are shifted before these multipliers with the amounts indicated by the pre-dividers used in the step-sizes computing module.

After this stage, the next intermediate terms are computed. The outputs of the previous multiplications are used as one set of inputs for the same two binary tree multipliers. The other set of inputs is represented by the results of the dividers $\mathbf{e}_0(n)/\det[\mathbf{N}(n)]$ and $\mathbf{e}_1(n)/\det[\mathbf{N}(n)]$, where $\det[\bullet]$ denotes the matrix determinant and $\mathbf{e}(n) = [\mathbf{e}_0(n), \mathbf{e}_1(n)]^T$. The computation of $\mathbf{N}^{-1}(n)$ is completed by these two divisions; the first set of inputs is shifted with the amount indicated by the pre-dividers used for these two dividers. The matrix product is

$$\begin{bmatrix} \mathbf{T}_1'(n) \\ \mathbf{T}_2'(n) \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{1,1}(n) & \mathbf{T}_{1,2}(n) \\ \mathbf{T}_{2,1}(n) & \mathbf{T}_{2,2}(n) \end{bmatrix} \left\{ \frac{1}{\det\left[\mathbf{N}(n)\right]} \begin{bmatrix} \mathbf{e}_0(n) \\ \mathbf{e}_1(n) \end{bmatrix} \right\}.$$

When the second stage is finished, the updating procedure is started. The memory blocks containing the far-end signal samples and the coefficients are read, and after another multiplication on the two binary tree multipliers, the final sum for update is made for each filter coefficient $\hat{h}_k(n)$, with $k = 0, 1, \ldots, L-1$, i.e.,

$$\hat{h}_k(n) = \hat{h}_k(n-1) + x(n-k)\mathbf{T}_1'(n) + x(n-k-1)\mathbf{T}_2'(n).$$

3.7 AEC block scheme

The AEC block scheme is generated on a modular basis. All the equations of the VSS-APA from Section 2 are implemented using the elementary modules described in the previous sub-sections. In addition, in order to provide synchronous operations, memory blocks are used to store the data, as it was previously mentioned. Figure 4 depicts the block scheme of the AEC, while Fig. 5 presents a timing diagram of the module functionality. For simplicity, the use of the two binary tree multipliers is reduced only to compute the estimated echo. However, their multiple functions are as described above.

The power estimates (see Fig. 4) are computed using sliding windows, as in (5). The value of the parameter λ is represented in 2's complement format. Consequently, the products that imply λ and $1 - \lambda$ are realized using shifts and summing units. A sequential multiplier is used for the square of the input signal. This sliding window is time-multiplexed for all the power estimates needed in (4).

3.8 Implementation results

The implementation targets a XC5VFX70T chip from Xilinx Virtex5 family [5] found on the evaluation board ML507 [6] from Xilinx. The synthesis results are obtained using Xilinx XST tool from Xilinx ISE 9.2i. The occupied area is reported in terms of slice components (each slice contains four flip flops and four 6-input LUTs). The proposed AEC implementation uses 4620 FFs (from a total of 44800), 5551 LUTs (from a total of 44800), and 3 BRAMs. The maximum frequency reported after placing and routing the design is 271.3 MHz. Starting from this speed result, as a future work, we can modify the timing diagram depicted in Fig. 5, in order to increase the reuse factor of the scheme components, especially for the binary tree multipliers. For maximum area optimization, we can imagine an AEC structure including only one multiplier and one fractional divider.

4. SIMULATION RESULTS

The functional and timing simulations are made using the ModelSIM 6.2g tool. We use a test bench that generates the echo by passing the far-end signal through a measured 512 tap acoustic impulse response (the sampling frequency is 8 kHz). The same length is used for the adaptive filter, i.e., $L = 512$. The far-end signal is a speech signal. The output of the echo path is corrupted by a white Gaussian noise with 20 dB signal-to-noise ratio (SNR). The performance measure is the normalized misalignment (in dB), defined as $20\log_{10}(\|\mathbf{h} - \hat{\mathbf{h}}(n)\|_2/\|\mathbf{h}\|_2)$, where $\mathbf{h}$ is the true impulse response of the echo path and $\|\bullet\|_2$ denotes the l_2 norm.

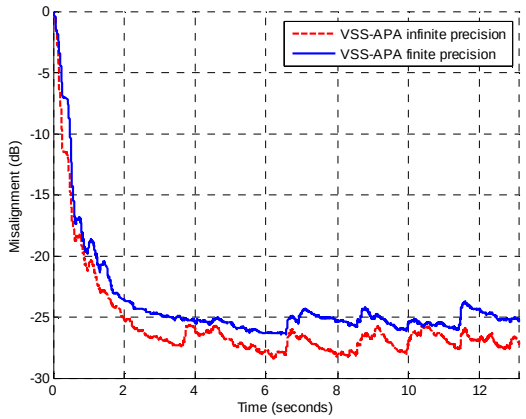

Figure 6 – Quantization effect on the misalignment of the VSS-APA. Single-talk case.

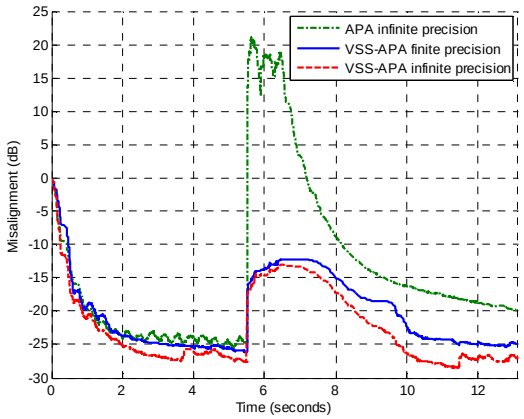

Figure 7 – Misalignment of the APA (infinite precision) and VSS-APA (finite and infinite precision). Double-talk case, without DTD.

In the first experiment, we test the AEC performance in a single-talk scenario. The effect of the quantization error on AEC performance is given in Fig. 6. Based on this simulation, we decided to use a 16 bit representation for the AEC inputs, and all the other variables (including the coefficients) being computed using 31 bits. These variables are used at full width in summing units and only on the first 16 most significant bits on multipliers. Figure 6 can be used as a reference when comparing the implementation performance with those described in [3], in infinite precision. In general, less than 2 dB degradation can be noticed.

In the second experiment, a double-talk scenario is considered. It is known that this is one of the most challenging situations in echo cancellation. In general, a double-talk detector (DTD) is required in this case, in order to control the algorithm's behaviour and to prevent its divergence. One of the main features of our VSS-APA is its robustness against double-talk [3]. In order to outline this aspect, we compare the VSS-APA with the classical APA (using a fixed step-size μ = 0.2), without using any DTD. The results are given in Fig. 7. First, the VSS-APA suffers less than 2 dB degradation when using finite precision implementation. Second, it is obvious that the VSS-APA is much more robust then APA during double-talk. As it was shown in [3], a simple Geigel DTD [8] improves this robustness for the VSS-APA, while the classical APA requires more sophisticated DTD, e.g., [9].

5. CONCLUSIONS

FPGA implementation of an AEC based on a VSS-APA was presented in this paper. The main features of the VSS-APA are its non-parametric nature and the moderate computational requirements. Nevertheless, several challenging operations are required in the step-size parameter formula, e.g., division, square-root, and matrix inversion. Dedicated blocks for these operations were implemented on the FPGA within the AEC scheme.

The provided AEC implementation represents a low-cost solution, which could be very attractive for real-world acoustic echo cancellation scenarios.

ACKNOWLEDGEMENTS

This work was supported by the UEFISCSU Romania under Grants PN-II-"Idei" no. 65/01.10.2007 and no. 331/01.10.2007.

The authors are thankful to Dr. Dennis R. Morgan from Bell Laboratories, Alcatel Lucent, for his valuable and constructive comments and suggestions.

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] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electron. Commun. Jpn.*, vol. 67-A, no. 5, pp. 19–27, May 1984.

[3] C. Paleologu, J. Benesty, and S. Ciochină, "A variable step-size affine projection algorithm designed for acoustic echo cancellation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 16, pp. 597–600, Nov. 2008.

[4] C. Anghel, C. Paleologu, J. Benesty, and S. Ciochină, "FPGA implementation of an acoustic echo canceller using a VSS-NLMS algorithm," in *Proc. IEEE ISSCS 2009*, pp. 369–372.

[5] "Xilinx Virtex 5 family user guide," www.xilinx.com.

[6] "Xilinx ML507 evaluation platform user guide," www.xilinx.com.

[7] J. Benesty, H. Rey, L. Rey Vega, and S. Tressens, "A nonparametric VSS NLMS algorithm," *IEEE Signal Process. Lett.*, vol. 13, pp. 581–584, Oct. 2006.

[8] D. L. Duttweiler, "A twelve-channel digital echo canceler," *IEEE Trans. Communications*, vol. 26, no. 5, pp. 647–653, May 1978.

[9] J. Benesty, D.R. Morgan, and J. H. Cho, "A new class of doubletalk detectors based on cross-correlation," *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 2, pp. 168–172, Mar. 2000.