On the FPGA Implementation of the VR-RLS Algorithms

Cristian Anghel, Silviu Ciochina
Telecommunications Department
University Politehnica of Bucharest
Bucharest, Romania
e-mail: {canghel, silviu}@comm.pub.ro

***Abstract*—This paper presents the main elements proposed for an efficient implementation on Field Programmable Gate Array (FPGA) of our novel Variable-Regularized Recursive Least Squares (VR-RLS) algorithm. The followed performance axes are the overall processing speed and the amount of used hardware resources. We also focus on this adaptive algorithm performance in the scenario of acoustic echo cancellation (AEC), from the finite precision implementation degradation point of view.**

Keywords- VR-RLS; FPGA; efficient implementation; adaptive algorithms

I. INTRODUCTION

Adaptive algorithms are very popular in many signal processing fields. One of the most known examples is the acoustic signal scenario, especially for the echo cancellation purposes. There are many studies in the literature referring to this topic. Our research team proposed in the last years several modified versions for the classic adaptive algorithms, pointing out the importance of variable step size (VSS) approach for the Least Mean Squares (LMS) family [1]-[4], respectively the variable regularized (VR) for Recursive Least Squares (RLS) ones [5]-[7]. The proposed algorithms were proved to be more robust from performance point of view on echo path change, double talk situations and noisy environments.

But, starting from these promising simulation results, obtained manly using Matlab, a question appeared: are these proposed algorithms stable enough when finite precision format is used in real implementation on digital signal processors (DSPs) or field programmable gate arrays (FPGAs)? We tried to answer to this question analyzing from theoretical point of view the quantization effect in [8][9]. More accurate results were presented in [10]-[13].

Starting from this previous experience, a new algorithm called Variable-Regularized Recursive Least Squares (VR-RLS) was proposed in [14]. The purpose of this paper is to present the main ideas used in order to obtain an efficient FPGA implementation of this algorithm. The efficiency is checked on two axes, one referring to the overall clock frequency and the other to the amount of used resources. The implementation targets a XC5VFX70 chip from Xilinx Virtex5 family [15] found on the evaluation board ML507 [16] from Xilinx. The synthesis results are obtained using Xilinx XST tool from Xilinx ISE 14.7.

The rest of this paper is organized as follows. Section II describes the equations belonging to VR-RLS algorithm. Section III describes the main proposed solutions for the hardware implementation. Section IV addresses the obtained results when synthesizing the very high speed description language (VHDL) source code. Section V highlights the conclusions of this paper.

II. VR-RLS ALGORITHM

Out of the four possible scenarios, one of the most common situations for an adaptive algorithm is the system identification problem. Figure 1 depicts this configuration in an acoustic echo canceller (AEC) context.

Considering the discrete time n, we can introduce the desired signal as:

$$d(n) = \mathbf{h}^T \mathbf{x}(n) + v(n) = y(n) + v(n), \qquad (1)$$

where $\mathbf{h} = \begin{bmatrix} h_0 & h_1 & \dots & h_{L-1} \end{bmatrix}^T$ is the impulse response of length L of the unknown system (that is to be identified), and superscript ^{T} denotes transpose of a matrix (or vector). The input vector is formed with the most recent L samples of the zero-mean input signal $x(n)$

$$\mathbf{x}(n) = \begin{bmatrix} x(n) & x(n-1) & \dots & x(n-L+1) \end{bmatrix}^T, \qquad (2)$$

and $v(n)$ is a zero-mean additive noise signal, which is independent of $x(n)$.

The goal for the configuration in Figure 1 is to estimate $\mathbf{h}$ with the adaptive filter $\hat{\mathbf{h}}(n) = \begin{bmatrix} \hat{h}_0(n) & \hat{h}_1(n) & \dots & \hat{h}_{L-1}(n) \end{bmatrix}^T$. In order to do this, a solution would be to use the cost function $J(n)$ corresponding to regularized least-squares criterion:

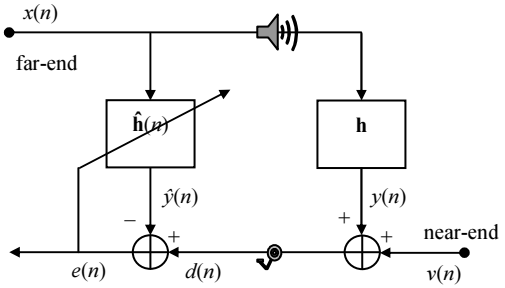

Figure 1. System model for acoustic echo cancellation.

$$J(n) = \sum_{i=0}^{n} \lambda^{n-i} \left[d(i) - \hat{\mathbf{h}}^T(n)\mathbf{x}(i) \right]^2 + \delta \left\| \hat{\mathbf{h}}(n) \right\|_2 \quad (3)$$

where $\lambda\left(0 < \lambda < 1\right)$ is the exponential forgetting factor, δ is the regularization parameter, and $\|\cdot\|_2$ is the ℓ_2 norm. Based on (3) it is shown in [14] that the update of the regularized RLS algorithm can be expressed as:

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \left[\hat{\mathbf{R}}_{\mathbf{x}}(n) + \delta \mathbf{I}_L \right]^{-1} \mathbf{x}(n)e(n), \quad (4)$$

where

$$\hat{\mathbf{R}}_{\mathbf{x}}(n) = \sum_{i=0}^{n} \lambda^{n-i} \mathbf{x}(i)\mathbf{x}^T(i) = \lambda \hat{\mathbf{R}}_{\mathbf{x}}(n-1) + \mathbf{x}(n)\mathbf{x}^T(n) \quad (5)$$

is an estimate of the correlation matrix of $\mathbf{x}\left(n\right)$, $\mathbf{I}_L$ is the identity square matrix (of size L), and the a priori error signal is given by:

$$e(n) = d(n) - \hat{y}(n) = d(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{x}(n). \quad (6)$$

Starting from this classic RLS algorithm described above, we introduced in [14] the VR-RLS form, which proposes a mean to find the regularization parameter δ.

If we consider the convergence of the adaptive filter (of course, keeping in mind that a certain misalignment will exist always), this will allow us to introduce the approximation:

$$y(n) \approx \hat{y}(n) \text{ and } \sigma_y^2 \approx \sigma_{\hat{y}}^2, \quad (7)$$

where $\sigma_u^2 = E\left[u^2(n) \right]$ is the variance of $u(n)$ (u being replaced here by d, y, v, $\hat{y}$), with $E[\cdot]$ denoting mathematical expectation. With these notations, and with the assumption that $y(n)$ and $v(n)$ are uncorrelated, we can write from (1) and (7)

$$\begin{aligned} \sigma_d^2 &= \sigma_y^2 + \sigma_v^2, \\ \sigma_v^2 &\approx \sigma_d^2 - \sigma_{\hat{y}}^2 \end{aligned} \quad (8)$$

For the power estimates a sliding window can be used as recursive computational method:

$$\hat{\sigma}_u^2(n) = \gamma \hat{\sigma}_u^2(n-1) + (1-\gamma)\hat{\sigma}_u^2, \quad u \in \{d, \hat{y}\} \quad (9)$$

where $\gamma = 1 - 1/(KL)$, with $K \geq 1$, and the initial values for the two power estimates from (9) are initialized with 0.

From (1) we can define the Signal to Noise Ratio (SNR)

$$\text{SNR} = \frac{\sigma_y^2}{\sigma_v^2}, \quad (10)$$

and from (9) we can rewrite an estimate of it as:

$$\hat{\text{SNR}}(n) = \frac{\hat{\sigma}_y^2(n)}{\left| \hat{\sigma}_d^2(n) - \hat{\sigma}_{\hat{y}}^2(n) \right|} \quad (11)$$

With this approach, following the demonstration from [14], the variable regularization parameter is obtained as:

$$\delta(n) = \frac{L\left[1 + \sqrt{1 + \hat{\text{SNR}}(n)} \right]}{\hat{\text{SNR}}(n)} \sigma_x^2 = \beta(n)\sigma_x^2, \quad (12)$$

where $\beta(n)$ is the ratio from (12) and it represents the variable normalized regularization parameter. Introducing (12) in (4), we obtain the VR-RLS algorithm, with update:

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \left[\hat{\mathbf{R}}_{\mathbf{x}}(n) + \delta(n)\mathbf{I}_L \right]^{-1} \mathbf{x}(n)e(n). \quad (13)$$

III. PROPOSED ARCHITECTURE

Let's evaluate now the algorithm described in the previous section from implementation complexity point of view. We consider the fractional 2's complement format, with variables on N bits, the bit N-1 indicating the sign. One can observe that complex operations, such as square root, high-order matrix inversion, fractional divider, and product with vector, are to be executed.

A. Fractional divider

There are several classic schemes for computing the fractional division. However, for fractional operations, it is very important to verify that the results are still in the accepted range of [-1, 1). This check has to be done also for division. If the *dividend* is bigger than the *divisor*, a *quotient* outside the range is obtained. For example, 0.8 divided by 0.5 equals 1.6. So we need a pre-divider module in order to make sure that we will have always the dividend less than the divider. If this is not the case, a certain number of shifts to the right will be applied to the dividend, until the condition becomes true. This number of shifts is counted and compensated afterwards, on another part of the algorithm. We consider a maximum possible number of shifts *lim*, obtained from Matlab simulations. This approach provides constant latency for this module.

On the other hand, if the dividend is less than the divisor, we may perform another action to improve the precision of the quotient. More precisely, usually we have the variables on a larger number of bits than needed. The most relevant example is the multiplication result: a number on N_a bits multiplied with a number on N_b bits will produce a result on N_a+N_b-1 bits. Since the multiplication appears periodically (with each new input sample), a truncation is needed after each such operation in order to keep the variables size constant. And since the product of two fractional numbers produces a result even smaller than the two operands, we can conclude that the truncation shall be carefully applied. In this context, considering that the dividend and the divider are obtain from such multiplication operations (for example)

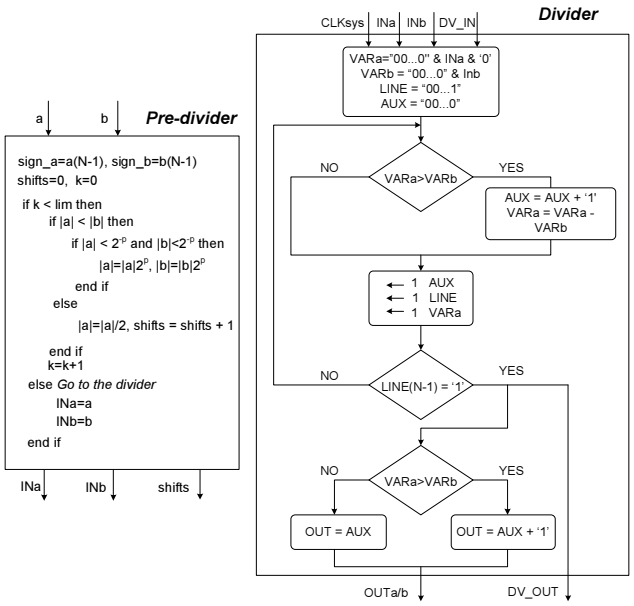

Figure 2. Pre-divider procedure and divider block scheme.

even if the dividend is less than the divider, a truncation applied to both of them before the division may lead to un-accurate results. So, we will check first if not both operands can be shifter a certain number of times to the left, and only after we truncate and then we divide. The complete idea is described in Figure 2, where a classic model of booth divider is also included.

B. Square root unit

The square root appears in (12), when computing the variable regularization parameter. In order to execute this operation, the approximation algorithm described in Figure 3 is used.

The algorithm 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 Figure 3).

```
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 3. Square-root approximation algorithm.

C. High-order matrix inversion

The last two most complex operations are in (5), respectively in (13).

In order to better understand (5), we may consider the first cases n=0, 1, 2 and 3 for a simplified scenario with L=3. One will observe that the matrix $\hat{\mathbf{R}}_{\mathbf{x}}(n)$ is symmetric, and moreover always the upper-left sub-matrix (L-1)x(L-1) from matrix $\hat{\mathbf{R}}_{\mathbf{x}}(n-1)$ is identical with the lower-right sub-matrix (L-1)x(L-1) from matrix $\hat{\mathbf{R}}_{\mathbf{x}}(n)$. In other words, it is enough to compute only the first column of the new matrix $\hat{\mathbf{R}}_{\mathbf{x}}(n)$ using the first column of matrix $\hat{\mathbf{R}}_{\mathbf{x}}(n-1)$:

$$\hat{\mathbf{R}}_{\mathbf{x}}^{(1)}(n) = \lambda \hat{\mathbf{R}}_{\mathbf{x}}^{(1)}(n-1) + \mathbf{x}(n)x(n) \qquad (14)$$

and then to obtain the complete matrix $\hat{\mathbf{R}}_{\mathbf{x}}(n)$, as shown in Figure 4.

The last and the most complex remaining operation is the high-order matrix inversion. Usually, L may be equal to 1024. This means that we have to compute the inverse of a matrix 1024 x 1024. This operation, besides the amount of required resources for processing, is also very time consuming. This is the reason why others alternative solutions were studied till now. One of the most efficient methods is the dichotomous coordinate descent (DCD) algorithm [17][18]. Our research team also obtained very good results in terms of FPGA implementation efficiency, the most important ones being presented in [8] and [10]. For this reason, we will not enter into details here about this already exposed solution.

IV. OBTAINED RESULTS

The proposed solutions described in the previous section fulfill both the requirements for high system clock frequency, respectively for reduced amount of used hardware resources. In order to show this, we propose an implementation on XC5VFX70 chip from Virtex 5 family. This FPGA has an architecture based on Configurable Logic Blocks (CLBs). Each such CLB contains 2 slices, one slice being formed of four flip-flops and four 6-inputs look-up tables.

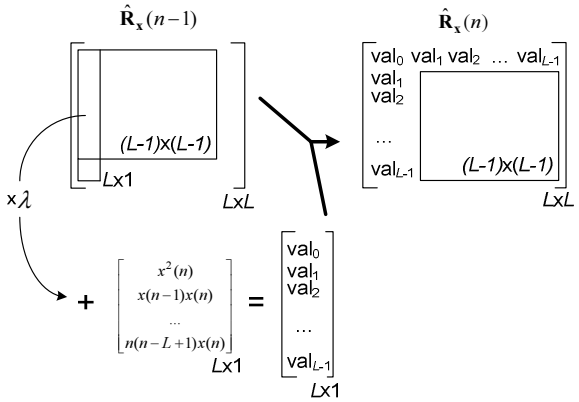

Figure 4. Matrix $\hat{\mathbf{R}}_{\mathbf{x}}(n)$ update.

The proposed AEC implementation, without the DCD part, uses 4620 flip flops (from a total of 44800), 5551 LUTs (from a total of 44800), and 3 block RAMs. The maximum frequency reported after placing and routing the design is 271.3 MHz. The results above were obtained when using a 16 bit representation for the AEC inputs, while 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 and dividers. This numerical format was selected based on the misalignment variation. The misalignment is defined as:

$$m(n) = 20\log_{10}\left(\left\|\mathbf{h} - \hat{\mathbf{h}}(n)\right\|_2 / \left\|\mathbf{h}\right\|_2\right) \quad (15)$$

and we accepted a degradation of maximum 2 dB between the 2 curves obtained in infinite precision, respectively in finite precision formats.

V. CONCLUSIONS

We presented in this paper the most important theoretical aspects related to VR-RLS algorithm. Starting from the obtained equations, and considering a fractional 2's complement numerical format, we identified the most complex operations. These were the divider, the square root, the matrix update and the matrix inversion. For each of them, an efficient solution from FPGA implementation point of view was proposed, except the matrix inversion, for which our research team proposed and presented previously an architecture based on the DCD algorithm.

The elements described in this paper can represent a solid ground for the efficient FPGA implementation of any adaptive algorithm.

ACKNOWLEDGMENT

The work has been funded by the Internal Research Grants Program offered by University Politehnica of Bucharest called "Excellency Research Grants UPB-EXCELENTA-2015".

REFERENCES

[1] C. Paleologu, J. Benesty, and S. Ciochina, "A variable step-size affine projection algorithm designed for acoustic echo cancellation," *IEEE Trans. Audio, Speech, Language Processing*, vol. 16, pp. 1466-1478, Nov. 2008.

[2] C. Paleologu, S. Ciochina, and J. Benesty, "Variable step-size NLMS algorithm for under-modeling acoustic echo cancellation," *IEEE Signal Processing Lett.*, vol. 15, pp. 5-8, 2008.

[3] S. Ciochina, C. Paleologu, and J. Benesty, "An optimized NLMS algorithm for system identification," *Signal Processing*, vol. 118, pp. 115-121, Jan. 2016.

[4] C. Paleologu, S. Ciochina, J. Benesty, and S. L. Grant, "An overview on optimized NLMS algorithms for acoustic echo cancellation," *EURASIP Journal Advances Signal Processing*, 2015, 2015:97 (19 pages).

[5] J. Benesty, C. Paleologu, and S. Ciochina, "On regularization in adaptive filtering," *IEEE Trans. Audio, Speech, Language Processing*, vol. 19, pp. 1734-1742, Aug. 2011.

[6] J. Benesty, C. Paleologu, and S. Ciochina, "Regularization of the RLS algorithm," *IEICE Trans. Fundamentals*, vol. E94-A, pp. 1628-1629, Aug. 2011.

[7] C. Paleologu, J. Benesty, and S. Ciochina, "A robust variable forgetting factor recursive least-squares algorithm for system identification," *IEEE Signal Processing Lett.*, vol. 15, pp. 597-600, 2008.

[8] C. Stanciu, C. Paleologu, J. Benesty, and S. Ciochina, "On a robust dual-path DCD-RLS algorithm for stereophonic acoustic echo cancellation," *Trans. Electronics and Communications*, vol. 58, pp. 9-14, Dec. 2013.

[9] C. Paleologu, S. Ciochina, and A. A. Enescu, "A family of recursive least-squares adaptive algorithms suitable for fixed-point implementation," *International Journal Advances in Telecommunications*, vol. 2, no. 2&3, pp. 88-97, 2009.

[10] C. Stanciu, C. Anghel, and L. Stanciu, " Efficient FPGA Implementation of the DCD-RLS Algorithm for Stereo Acoustic Echo Cancellation," *2015 International Symposium on Signals, Circuits and Systems (ISSCS)*, Iasi, 2015, pp. 1-4. doi: 10.1109/ISSCS.2015.7204008

[11] C. Stanciu, C. Anghel, C. Paleologu, J. Benesty, F. Albu, and S. Ciochina, "FPGA implementation of an efficient proportionate affine projection algorithm for echo cancellation," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2011, pp. 1284-1288, Barcelona, Spain.

[12] C. Anghel, C. Paleologu, J. Benesty, and S. Ciochină, "FPGA Implementation of a Variable Step-Size Affine Projection Algorithm for Acoustic Echo Cancellation", in *Proc. European Signal Processing Conference (EUSIPCO)*, 2010, pp 532-536, Aalborg, Denmark

[13] C. Anghel, C. Paleologu, J. Benesty, and S. Ciochină, "FPGA Implementation of an Acoustic Echo Canceller Using a VSS-NLMS Algorithm", in *Proc. IEEE International Symposium on Signals, Circuits and Systems* (ISSCS), 2009, pp. 369-372, Iasi, Romania.

[14] C. Stanciu, C. Iliescu, C. Paleologu, J. Benesty, C. Anghel, "Robust Variable-Regularized RLS Algorithms", in Proc IEEE HSCMA, March 2017, San Francisco, USA, pp. 171-175

[15] "Xilinx Virtex 5 family user guide," www.xilinx.com, retrieved: February, 2017

[16] "Xilinx ML507 evaluation platform user guide," www.xilinx.com retrieved: March, 2017

[17] J. Liu and Y. Zakharov, "Dynamically regularized RLS-DCD algorithm and its FPGA implementation", Asilomar Conference on Signals, Systems and Computers, 2008, pp. 1876-1880

[18] Z. Quan, Y. Zakharov, J. Liu, "DCD-based simplified matrix inversion for MIMO-OFDM", IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp. 2389-2392

 ISBN: 978-1-61208-546-3