DESIGN OF LRLS ADAPTIVE FILTER WITH LOW ADAPTATION DELAY IN EFFICIENT MODE

1 J Stalone Marshall, 2 Dr. A. Rajaram
1 PG Scholar, Department of Electronics and Communication, Karpagam University, India, Marzie2007@hotmail.com.
2 Assistant Professor, Department of Electronics and Communication, Karpagam University, India.

ABSTRACT: In this paper, we present an efficient architecture for the implementation of a normalized lattice recursive least square adaptive filter. The adaptive process involves the use of a cost function, which is a criterion for optimum performance of the filter, to feed an algorithm, which determines how to modify filter transfer function to minimize the cost on the next iteration. For achieving lower adaptation delay and area, delay power implementation, we use a partial product generator and a super scalar concept which uses an optimized balanced structure across the time-consuming combinational blocks of the proposed structure. For synthesis, we find that the proposed structure allows nearly 19% less area and the power consumption is reduced to 15%, than the convention least mean square adaptive filter. We propose an efficient fixed point implementation structure of the proposed architecture and have derived an expression for steady-state error. We have shown that the steady-state error obtained from the theoretical result matches with the simulated result. We also have proposed a variable filter structure, provides 19% savings in the total power and 10% saving in area before pruning without compromising the steady-state error performance.

Index Term: Lattice recursive least square (LRLS) algorithm, Floating Point arithmetic operation, Circuit organization and Adaptive filter.

I. INTRODUCTION

Recursive least squares (RLS) was discovered by Gauss but lay unused or ignored until 1950 when Plackett rediscovered the original work of Gauss from 1821. In general, the RLS can be used to solve any problem that can be solved by adaptive filters.

The Recursive least squares (RLS) adaptive filters is an algorithm which recursively finds the filter coefficients that minimize a weighted linear least squares cost function relating to the input signals. This is in contrast to other algorithms such as the least mean squares (LMS) that aim to reduce the mean square error. In the derivation of the RLS, the input signals are considered deterministic, while for the LMS and similar algorithm they are considered stochastic.[1],[2] Compared to most of its competitors, the RLS exhibits extremely fast convergence. However, this benefit comes at the cost of high computational complexity. Since the conventional RLS algorithm does not support super scalar concept because of its recursive behaviour, it is modified to a form called the lattice recursive least square algorithm (LRLS) [3],[4] which supports super scalar concept which allows super scalar concept implementation of the filter.

The Modified LRLS adaptive filter is related to the standard RLS except that it requires fewer arithmetic operations (order N). It offers additional advantages over conventional LMS algorithms such as faster convergence rates, modular structure, and insensitivity to variations in Eigen-value spread of the input correlation matrix. The LRLS algorithm described is based on a posterior errors. A lot of work has been done to implement the LRLS algorithm in systolic architecture to increase the maximum usable frequency, but they have complex adaptation delay, which is quite high for large-order filter. In the next section, we review the Modified LRLS algorithm, and in section 3, we describe the proposed optimized architecture for its implementation. Section 4 deals with the fixed point implementation and the synthesis of the proposed architecture and the comparison with the existing architecture. Conclusion is given in section 5.
II. RELATED WORK

Review Of The Modified Lattice Recursive Least Square Algorithm

The Lattice Recursive Least Squares adaptive filter is related to the standard RLS except that it requires fewer arithmetic operations (order N). It offers additional advantages over conventional LMS algorithms such as faster convergence rates, modular structure, and insensitivity to variations in eigenvalue spread of the input correlation matrix. The LRLS algorithm described is based on a posterior errors and includes the normalized form. The derivation is similar to the standard RLS algorithm and is based on the definition of $d(k)$. In the forward prediction case, we have $d(k) = x(k)$ with the input signal $x(k - 1)$ as the most up to date sample. The backward prediction case is $d(k) = x(k - i - 1)$, where i is the index of the sample in the past we want to predict, and the input signal $x(k)$ as the most recent sample.

![Fig 1. Structure of the Existing System](image1)

![Fig 2. Structure of the Modified LRLS](image2)

The idea behind RLS filters is to minimize a cost function $C$ by appropriately selecting the filter coefficients $W_n$, updating the filter as new data arrives. The error signal $e(n)$ and desired signal $d(n)$ are defined in the negative feedback diagram above, the error implicitly depends on the filter coefficients through the estimate $\hat{d}(n)$.

$$e(n) = d(n) - \hat{d}(n) \quad (1)$$
The weighted least squares error function $C^r$—the cost function we desire to minimize being a function of $e(n)$ is therefore also dependent on the filter coefficients:

$$ C(w_n) = \sum_{i=0}^{n} \lambda^{n-i} e^2(i) \quad (2) $$

Where $0 < \lambda \leq 1$ is the "forgetting factor" which gives exponentially less weight to older error samples. Where the input vector $X_n$ and the weight vector $W_n$ at the nth iteration, $d(n)$ is the desired response, $Y_n$ is the filter output and $N$ is the number of weights used in Modified LRLS adaptive filter.

The cost function is minimized by taking the partial derivatives for all entries $k$ of the coefficient vector $W_n$ and setting the results to zero

$$ \frac{\partial C(w_n)}{\partial w_n(k)} = \sum_{i=0}^{n} 2\lambda^{n-i} e(i) \frac{\partial e(i)}{\partial w_n(k)} = \sum_{i=0}^{n} 2\lambda^{n-i} e(i) x(i-k) = 0 \quad k = 0,1,...,p \quad (3) $$

Next, replace $C(n)$ with the definition of the error signal

$$ \sum_{i=0}^{n} \lambda^{n-i} \left[ d(i) - \sum_{l=0}^{n} w_n(l) x(i-l) \right] x(i-k) = 0 \quad k = 0,1,...,p \quad (4) $$

$$ R_x(n) w_n = r_{dx}(n) \quad (5) $$

where $R_x(n)$ is the weighted sample correlation matrix for $x(n)$, and $r_{dx}(n)$ is the equivalent estimate for the cross-correlation between $d(n)$ and $x(n)$. Based on this expression we find the coefficients which minimize the cost function as

$$ w_n = R_x^{-1}(n) r_{dx}(n) \quad (6) $$

The adaptive filter with different $n_1$ and $n_2$ are simulated for a system identification problem. In order to generate the coefficient vector we are interested in the inverse of the deterministic autocorrelation matrix

$$ R_x^{-1}(n) = \left[ \lambda R_x(n-1) + x(n)x^T(n) \right]^{-1} \quad (7) $$

The second step follows from the recursive definition of $r_{dx}(n)$. Next we incorporate the recursive definition of $P(n)$ together with the alternate form of $g(n)$ and get

$$ w_n = \left[ \lambda P(n-1) + g(n)g^T(n) \right]^{-1} \quad (8) $$

$$ \Delta w_{n-1} = g(n) \alpha(n) \quad (9) $$
This intuitively satisfying result indicates that the correction factor is directly proportional to both the error and the gain vector, which controls how much sensitivity is desired, through the weighting factor $\lambda$.

III. PROPOSED ARCHITECTURE

As shown in fig.2, there are two main computation block in the adaptive filter: 1) variable filter block, and 2) update algorithm block. In this section we discuss the design strategy for designing the proposed structure to minimise the adaptation delay in the variable filter block, followed by the update algorithm block. Fig.3 shows the NLRLS schematic view and the structure of the proposed view in a Cadence view.

![NLRLS schematic](image)

**Fig.3. NLRLS schematic**

**Hypotheses Estimation:**

Despite the possibility to implement the RLS estimation by formulas introduced in Section 2, it is more convenient to use one of the state-of-the-art RLS algorithms. As the most convenient algorithm for implementation, the recursive least-squares lattice [4] in the error-feedback form was chosen. As suggested in [17], the normalized a posterior errors are used to reduce the complexity of the algorithm. The computational complexity of this algorithm is $24N$, where $N$ denotes the filter order (dimension). As mentioned above, the estimation of probability of hypotheses $h$ by (1) requires performing one RLS estimation for each hypothesis $h$. Thus, the NM-array of RLS filters has to be calculated. The most important property making the RLS lattice suitable for the hypotheses estimation is its modular structure. The RLS lattice filter consists of a cascade of identical modules.

Each module implements the order update, which means that it is using $i$th order output from the preceding module and increases the order of estimation to $i + 1$. Consequently, estimations of all orders up to $N$ can be found during computations. Using this principle, the number RLS filters required for the hypotheses estimation can be reduced to $M$.

In our solution, the evaluation of probability estimates is divided into two stages. The first stage performs the order update, which uses the “old” probability estimates and updates them by new data. This operation is represented by the numerator of (1). In the second stage, the normalization of the updated order estimates is performed. The normalization is represented by the denominator of (1).
The forgetting on hypotheses pdf is applied in the normalization stage. The order update can be integrated into the RLS lattice algorithm. The RLS lattice algorithm with order and forgetting factor update is summarized in Algorithm 1. For the illustration the update of estimates to order \( i + 1 \) from order \( i \) is depicted in Figure 1. The algorithm presented in Algorithm 1 is in the form, where input \( u \) is used to estimate desired value \( d \). For identification of one-dimensional auto regression model the input must be connected as presented in Figure 2. Then, the probabilistic approach given in Section 2 can be used for estimation of order and forgetting factor probability as also shown in the figure. The RLS lattice algorithm parameters are summarized in Table 1. For the probability \( p(h_n|D_n) \), \( \phi_n = \pi_i,\lambda_n \) will be further used as a more simple notation, where \( h \) was defined as the ordered pair

\[
(i, \lambda), \ i \in \{0, \ldots, N\}, \ h \in \{1, \ldots, M\}.
\]

Before the first iteration of the algorithm, initial hypotheses pdf has to be set and the look-up tables have to be initialized. The initial hypotheses pdf can be selected as

\[
\pi_i,\lambda_{-1} = \frac{1}{(N + 1)M} \forall i, \lambda,
\]

where \( \pi_i,\lambda \) is the probability of order \( i \) and forgetting factor \( \lambda \) at time \( n \).

Adding the order and forgetting estimation, the original RLS lattice algorithm increases its complexity to \( 31N \). Thus, the number of operations for maintaining \( N +1 \) order and \( M \) forgetting factor hypotheses is \( 31NM \). The normalization of updated probabilities requires \( 2M(N +1)+M−1 \) operations. Considering these figures, we can state the complexity of the RLS lattice with the estimation of hypotheses which is \( 33NM + 3M - 1 \) operation, provided that the division of two powers is regarded as one operation. It is evident that the implementation of M RLS lattice estimations to test each hypothesis can be easily parallelized.

For each forgetting factor hypothesis, one RLS lattice instance extended by the probability update can be evaluated in parallel. When all filters have been calculated, the normalization is performed before new data are acquired. Such an arrangement can be efficiently implemented in FPGAs. Optimized RLS lattice core the result of RLS lattice implementation is a standalone IP core. The internal variables of the core are expected to be stored in external memories. It allows more convenient integration of the core into the soft core processor.

A special memory organization, grouping the instances of one internal variable into one distributed memory element, was used. It allows us to reduce the size of multiplexer connected to the shared arithmetic unit macros, which is demonstrated in Figure 5. The experiments show that 8% of FPGA resources were saved by this approach. Integration to a microprocessor system considering common solutions, where algorithm is used as an IP core in one purpose hardware design, our aims to provide a versatile configurable RLS lattice solution. Thus, the optimized RLS lattice core was integrated as a coprocessor to the Microblaze processor system. For its development, the Xilinx system generator (XSG) was used. It is possible to create the coprocessor PCORE, which can directly be integrated to the Microblaze system. Such solution provides maximal applicability of the RLS lattice core, although at the expense of slight performance loss against one purpose design based on the same core. The XSG schematic of the coprocessor can be seen in Figure 6. It consists of the RLS lattice core denoted LSL, which can also be used as a standalone core of the input and output data buffers denoted in Buf and output Buf, of the control word sb2hc control and hc2mb status, and of the communication interface denoted Comm.

The Common block is responsible for the batch processing of input data by the RLS lattice core in the LSL block. It stores filter results to the output buffer. It is capable of working with the RLS lattice core containing one up to four filter instances. The LSL module consists of order updating loop (see algorithm in Algorithm 1) with the input consisting of four values. After increasing the estimation to higher order, the same four data can be regarded as output. Otherwise, only the internal states are
altered. As a consequence, one filter can use another filter output as its input and to continue in the computation of order updates.

The solution of higher order filter can be obtained using the pipelined solution consisting of multiple RLS lattice filter instances. That is why the Comm block in the coprocessor design has to be capable of reconfiguring data path between RLS lattice instances, in order to use them either as parallel four channel RLS lattice filter (each with order up to 126), or it is possible to connect up to four instances serially and to create pipelined RLS lattice solution with order up to 504, achieving 4x higher performance. It is the reason why, in Figure 4 the fifo block storing the intermediate results is used in the pipelined organization of the coprocessor.

**Dynamic Reconfiguration**

The implementation of the RLS lattice as a coprocessor connected to the Microblaze makes possible to use the dynamic configuration for loading and unloading the coprocessor while the microcontroller is running.

Fig.4. NLRLS device view schematic

The loading of the coprocessor can be initiated on demand. The software version of the RLS lattice filter was developed. The floating-point number representation is used in the Microblaze, whereas the 19-bit LNS are used in hardware.

The software conversions based on the HSLA library were used for implementation of migration between hardware and software. The conversions are based on evaluation of base-2 logarithm contained in the Microblaze library. Such conversions are time consuming even if a hardware support of floating-point is included in the Microblaze.

To control the load of the processor and the power consumption, a mechanism for migration the task from the processor to coprocessor was developed. The software version of the RLS lattice runs in the Microblaze when no other tasks require using the processor. When there is a need to use the Microblaze for other tasks, the processor is freed and the RLS lattice is run in the coprocessor. The available run-time configurations are presented in Figure 8. One is the software solution, remaining three are the coprocessor versions containing one to four RLS lattice filters.
IV. SIMULATION RESULTS

In the simulation the reference input signal \( x(n) \) is a white Gaussian noise of power two-dB generated using random function in MATLAB, the desired signal \( d(n) \), obtained by adding a delayed version of \( x(n) \) into clean signal \( s(n) \), \( d(n) = s(n) + x_1(n) \) as shown in Fig. 2. Fig. 2 Clean tone (sinusoid) signal \( s(n) \); (b) Noise signal \( x(n) \); (c) Delayed noise signal \( x_1(n) \); (d) desired signal \( d(n) \). The simulation of the LMS algorithm is carried out with the following specifications:

Filter order \( N=19 \), step size \( \mu=0.001 \) and iterations= 8000. The LMS filtered output is shown in Fig. 3 (a), the mean squared error generated as per adaption of filter parameters is shown in Fig. 3. The step size \( \mu \) control the performance of the algorithm, if \( \mu \) is too large the convergence speed is fast but filtering is not proper, if \( \mu \) is too small the filter gives slow response, hence the selection of proper value of step-size for specific application is prominent to get good results.

The effect of step size on mean squared error is illustrated in Fig. 7. Fig. 4 and Fig. 5 shows the output results for NLMS and RLS algorithms respectively. If we investigate the filtered output of all
algorithms, LMS adopt the approximate correct output in 2800 samples, NLMS adopt in 2300 samples and RLS adopt in 300 samples. This shows that RLS has fast learning rate. The filter order also affect the performance of a noise cancellation system. Fig. 6 illustrate how the MSE change as we change filter order, when filter order is less (<15) LMS has good MSE as compared to NLMS and RLS but as the filter order increased (>15) the performance of RLS becomes good and LMS has poor performance it proves that the selection of right filter order is necessary to achieve the best performance.

In our work the appropriate filter order is 19 therefore all simulations are carried out at N=19. In table 1 performance analysis of all three algorithms is presented in term of MSE, percentage noise reduction, computational complexity and stability [9]. It is clear from the table 1, the computational complexity and stability problems increases in an algorithm as we try to reduce the mean squared error. NLMS is the favourable choice for most of the industries due less computational complexity and fair amount of noise reduction.
This section presents the results of simulation using MATLAB to investigate the performance behaviours of various adaptive filter algorithms in non-stationary environment with two step sizes of 0.02 and 0.004. The principle means of comparison is the error cancellation capability of the algorithms which depends on the parameters such as step size, filter length and number of iterations.

A synthetically generated motion artefacts and power line interference are added with respiratory signals. It is then removed using adaptive filter algorithms such as LMS, Sign LMS, Sign-Sign LMS, Signed Regressor, BLMS and NLMS. All Simulations presented are averages over 1000 independent runs. Removal of Motion Artefacts Respiratory signal is represented by second-order autoregressive process that is generated according to the difference equation, \( x(n) = 1.2728x(n-1) - 0.81x(n-2) + v(n) \) (9) where \( v(n) \) is randomly generated noise. Figure 2 and Figure 3 shows the convergence of filter coefficients and Mean squared error using LMS and NLMS algorithms.

An FIR filter order of 32 and adaptive step size parameter (\( \mu \)) of 0.02 and 0.004 are used for LMS and modified step sizes (\( \beta \)) of 0.01 and 0.05 for NLMS. It is inferred that the MSE performance is better for NLMS when compared to LMS. The merits of LMS algorithm is less consumption of memory and amount of calculation. Removal of Power line Interference. A synthetic power line interference of 50 Hz with 1mv amplitude is simulated for PLI cancellation.

Power line interference consists of 50 Hz pickup and harmonics which can be modelled as sinusoids and combination of sinusoids. Figure 4 shows the generated power line interference. The mean square learning curves for various algorithms are depicted as shown in Figure 5. The input \( x(n) \) is 0.18 Hz sinusoidal respiratory signal. It is observed that minimization of error is better with BLMS compared with other algorithms.
V. CONCLUSION

This study has revealed useful properties of various adaptive filter algorithms. The objective is to optimize different adaptive filter algorithms so that we can reduce the MSE so as to improve the quality of eliminating interference. It is inferred that the MSE performance is better for NLMS when compared to LMS. The merits of LMS algorithm is less consumption of memory and amount of calculation. It has been found that there will be always trade off between step sizes and Mean square error. It is also observed that the performance depends on the number of samples taken for consideration. Choosing an algorithm depends on the parameter on which the system has much concern. The future work includes the optimization of algorithms for all kinds of noises and to use the optimized one in the implementation of DSP Microcontroller that estimates the respiratory signal.

In this work, different Adaptive algorithms were analyzed and compared. These results shows that the LMS algorithm has slow convergence but simple to implement and gives good results if step size is chosen correctly and is suitable for stationary environment. For a lower filter order (<15) the LMS proved to have the lowest MSE then the NLMS and RLS, but as we increase the filter order (>15) the results showed the opposite, so we need to take this in consideration when selecting the algorithm for a specific application. When input signal is non-stationary in nature, the RLS algorithm proved to have the highest convergence speed, less MSE, and highest percentage of noise reduction but at the cost of large computational complexity and memory requirement. The NLMS algorithm changes the step-size according to the energy of input signals hence it is suitable for both stationary as well as non-stationary environment and its performance lies between LMS and RLS. Hence it provides a trade-off in convergence speed and computational complexity. The implementation of algorithms was successfully achieved, with results that have a really good response.

REFERENCES


AUTHORS PROFILE

Stalone Marshall,J received the BE Degree from the Department of Electronics and Communication in Sudharsan Engineering College,Padukottai affiliated by Anna university,Chennai,India, in 2011.His Research interest includes design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal, image and video processing, ASIC design, Control systems,vlsi design and
algorithms for low-power/low-area/high-performance digital signal processing, ASIC design, Control systems, Sensor Networks.

Rajaram A received the BE degree in electronics and communication Engineering from the Govt., College of Technology, Coimbatore, Anna University, Chennai, India, in 2006, the ME degree in electronics and communication engineering (Applied Electronics) from the Govt., college of Technology, Anna University, Chennai, India, in 2008 and he received the Ph.D. degree in electronics and communication engineering from the Anna University of Technology, Coimbatore, India in March 2011. He is currently working as a Associate Professor, ECE Department in Karpagam University, Coimbatore, India. His research interests include mobile adhoc networks, wireless communication networks (WiFi, WiMax HighSlot GSM), novel VLSI NOC Design approaches to address issues such as low-power, cross-talk, hardware acceleration, Design issues includes OFDM MIMO and noise Suppression in MAI Systems, ASIC design, Control systems, Fuzzy logic and Networks, AI, Sensor Networks.