# Pipelining Extended Givens Rotation RLS Adaptive Filters

*Shing Tenqchen[1&**], Ji-Horn Chang*, Wu-Shiung Feng* and Bor-Sheng Jeng***

[1]Lab 331, Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.

*Department of Electronics Engineering, Chang Gung University, Taoyuan, Taiwan, R.O.C.

**Chunghwa Telecom Telecommunication Labs., 12, Lane 551, Sec. 5,

Min-Tsu Rd., Yang-Mei Zien, Tao-Yuan County, Taiwan 326, R.O.C.

E-mail: d86028@cad.ee.ntu.edu.tw and stc@cht.com.tw

**ABSTRACT -**

In this paper, we propose a new pipelining extended Givens Rotation Recursive Least Square (PEGR-RLS) architecture using look-ahead technique. The square-root-free forms of QRD-RLS are also difficult to pipeline. The PEGR-RLS algorithm (referred to as Scaled Tangent Rotation, STAR-RLS) is designed such that fine-grain pipelining can be accomplished with little hardware overhead. Similar to STAR-RLS, this algorithm is not exactly orthogonal transformations but tends to become orthogonal asymptotically. This algorithm also preserves the desired properties of the STAR-RLS algorithm. Specifically, it can be pipelined at very low forgetting factor by using extended look-ahead. Simulation results are presented to compare the performance of the STAR-RLS, QRD-RLS, and LMS algorithms.

## 1. Introduction

Adaptive filtering has wide applications in channel equalization, system identification, Beamforming, and image processing. A typical channel equalizer scheme is shown in Fig. 1. The idea of systolic arrays was developed by Kung and Leiserson (1978) [1] and with its new and exciting avenue for matrix-oriented inversion.

Here, a desired signal *d(n)* is transmitted over a noisy channel. The goal of the channel equalizer is to recover *d(n)* from *u(n)*. This is done by passing the received signal through a filter whose weights are determined by the adaptive filter. During the training phase, both *d(n)* and *u(n)* are known. An adaptive algorithm is employed to adapt the filter weights to the channel characteristics. After the training phase, the weights converge to the channel characteristics and the weights can then be used to process the actual data. The QRD-RLS and STAR-RLS algorithms [2]-[5] are the most promising RLS algorithms since it is known to have very good numerical properties and can be mapped to a pipelined systolic array. This algorithm is, hence, very suitable for VLSI implementation.

- This work was supported by National Science Council under NSC: 88-2216-E-002-018.

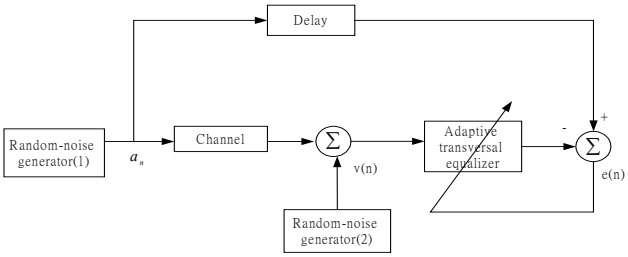



Fig. 1. Adaptive channel equalizer scheme in digital communication.

The speed of the algorithm is, however, limited by the time required by the individual cells. The computation of each cell contains recursive operations, and hence, the operations within a cell cannot be pipelined, i.e., finegrain pipelining cannot be achieved in this case. To increase the speed of the QRD-RLS, we could use the lookahead method [6], [7] or block processing techniques [8]. Using lookahead in QRD-RLS results in large hardware overhead. Consequently, this technique is not practical for the QRD-RLS algorithm. Block processing was used to speed up the QRD-RLS in [8]. However, the hardware will increase linearly with speedup and, hence will be expensive. There are square-root free forms of QRD-RLS, which are more computationally efficient than the original algorithms (see [10]-[11]. Recently, other fast QRD-RLS algorithm based on Givens rotations was introduced in [12]. These algorithms also have the same pipelining difficulty as the QRD-RLS algorithm. Thus, our aim is to pipeline the RLS algorithm with little increase in hardware for all of the forgetting factors. The overhead due lookahead has some disadvantages in hardware implementation has already be discussed before mentioned. Thus, we use delay relaxation to overcome the overhead of recursive loops in systolic array architecture implemented PEGR-RLS algorithm. The price paid for this is a limited loss of convergence rate depending on the speedup used.

## 2. Systolic-Array-Based Recursive Least Square Algorithms

### *A. Background and Notation*

In least square (LS) filtering, the desired response *d(n)* is estimated as the weighted sum of the present sample and a

few of the past samples. The input data is a 1×M row vector whose individual entries denoted by $\mathcal{U}(i)$

$$U(i) = [u_0(i) \quad u_1(i) \quad u_2(i) \quad \ldots \quad u_{k-2}(i) \quad u_{k-1}(i)] \tag{1}$$

and a data matrix $\mathcal{A}(n)$

$$\mathcal{A}(n) = [\mathcal{U}(1) \quad \mathcal{U}(2) \quad \ldots \quad \mathcal{U}(n)]^T = \begin{bmatrix} u_0(1) & u_1(1) & \cdots & u_{\mathcal{K}-1}(1) \\ u_0(2) & u_1(2) & \cdots & u_{\mathcal{K}-1}(2) \\ \vdots & \vdots & \ddots & \vdots \\ u_0(n) & u_1(n) & \cdots & u_{\mathcal{K}-1}(n) \end{bmatrix} \tag{2}$$

Consider also a known column vector $\overline{w}$ and a positive-definite weighting matrix $\Pi_0$. The objective is to determine an $M$×1 column vector $w$, also known as the weight vector, so as to minimize the weighted error sum:

$$E(n) = (w(n) - \overline{w}(n))^T \left[\lambda^{-(N+1)} \Pi_0\right]^{-1} (w(n) - \overline{w}(n)) + \sum_{i=0}^{N} \lambda^{N-i} \left| d(i) - u_i^T(i) w(i) \right|^2 \tag{3}$$

where λ is a positive scalar that is less than or equal to one $(0 << \lambda \le 1)$. It is often called as forgetting factor since past data is exponentially weighted less than the more recent data. The special case λ = 1 is known as the growing memory case, since, as the length $N$ of the data grows, the effect of the pass data is not attenuated. The error vector *e(n)* is expressed as

$$\varepsilon(n) = d(n) - y(n) = d(n) - u^T(n) w(n) \tag{4}$$

where ***w(n)*** is the desired weight vector and *d(n)* is the desired response vector, defined similar to (1). The weight vector ***w(n)*** is chosen so as to minimize the index of performance *E(n),* where

$$E(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 = \varepsilon^T(n) \Lambda(n) \varepsilon(n) = \left\| \Lambda^{1/2}(n) \varepsilon(n) \right\|^2 \tag{5}$$

where $\lambda$ is an exponential weighting forgetting factor and $\Lambda(n)$ is an *n*×*n* diagonal exponential weighting matrix defined by

$$\Lambda(n) = diag[\lambda^{n-1} \quad \lambda^{n-2} \quad \cdots \quad 1] = \begin{bmatrix} \lambda^{n-1} & 0 & \cdots & 0 \\ 0 & \lambda^{n-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

It then followed that

$$E(n) = (w(n) - \overline{w}(n))^T \left[\lambda^{-(N+1)} \Pi_0\right]^{-1} (w(n) - \overline{w}(n)) + \left\| \Lambda_N^{1/2} \xi(N) \right\|_2^2. \tag{7}$$

Then, the optimal solution is obtained by solving the known normal linear system of equations [14],

$$R(n) w_o(n) = p(n) \tag{8}$$

where $R(n)$ is the correlation matrix of the tap input vector, $w_o(n) = w(n) - \overline{w}(n)$, and $R(n)$ defined as

$$R(n) = \sum_{i=1}^{n} \lambda^{N-i} U(i) U^T(i). \tag{9}$$

$p(n)$ is the data correlation between data vector and the desired signal, given by

$$p(n) = \sum_{i=1}^{n} \lambda^{N-i} U(i) d(i). \tag{10}$$

Exploiting the matrix inversion lemma, the RLS algorithm can be computed using the following state-space equations:

$$x(n+1) = \lambda^{-1/2} x(n) \tag{11a}$$

$$y(n) = u^T(n) x(n) + \nu(n) \tag{11b}$$

with

$$x(0) = w(0), \ \overline{x}(0) = \overline{w}(0), \ \mathrm{cov}(x(0)) = \lambda^{-1} \Pi_0, \tag{12}$$
$$E(v(i) v(j)) = \delta_{ij}.$$

This is the special case of Kalman filter [8], with the following quantities,

$$y(n) = \frac{d(n)}{(\sqrt{\lambda})^n}, \ x(n) = \frac{w(n)}{(\sqrt{\lambda})^n}, \tag{13}$$
$$x(0) = w(0), \ \overline{x}(0) = \overline{w}(0).$$

With the RLS problem we also related to Kalman filter innovation *e*(*n*), which is defined by *e*(*n*) = *y*(*n*) – *u*(*n*)*w*(*n-1*). Its variance is denoted by $\gamma(n)$. The *a priori* error $\xi_M(n)$, and the *a posteriori* error, $e_M(n)$ are denoted as

$$\xi_M(n) = d(n) - u^T(n) w(n-1)$$
$$e_M(n) = d(n) - u^T(n) w(n),$$

respectively. Then, the RLS error can be expressed by

$$e(n) = y(n) - u^T(n) \hat{x}(n-1) = \frac{1}{(\sqrt{\lambda})^n} [d(n) - u^T(n) w(n-1)] = \frac{1}{(\sqrt{\lambda})^n} \xi_M(n)$$

$$e_M(n) = d(n) - (\sqrt{\lambda})^{n+1} u^T(n) \hat{x}(n) = \left[1 - \frac{u(n) P(n) u^T(n)}{1 + u(n) P(n) u^T(n)}\right] \xi_M(n)$$
$$= \beta^{-1}(n) \xi_M(n)$$

This means that the so-called conversion factor, $\beta^{-1}(n)$, that converts the *a priori* error $\xi_M(n)$ to the *a posteriori* error $e_M(n)$.

### *B. The square-root QRD-RLS Algorithm*

The QRD-RLS is known to have excellent numerical properties due to the choice of the weight vector *w(n)* to minimize the cost function $\mathcal{E}(n)$. It is important to note that the data matrix $\mathcal{A}$(n) is always assumed to have full column rank. The QRD-RLS algorithm is summarized here.

1) For $n > m$, perform the following update:

$$\begin{bmatrix} R(n) & p(n) & R^{-1}(n) u(n) \\ 0^T & \varepsilon(n) & \gamma^{1/2}(n) \end{bmatrix} = Q(n) \begin{bmatrix} \lambda^{1/2} R(n-1) & \lambda^{1/2} p(n-1) & 0 \\ u^T(n) & d(n) & 1 \end{bmatrix} \tag{14}$$

where $Q(n)$ is an *n*×*n* orthogonal matrix, $R(n)$ is the *m*×

$m$ upper triangular matrix resulted from QR decomposition, $p(n)$ is a $m \times 1$ vector. Angle normalized estimation error is denoted by $\varepsilon(n)$ and the corresponding conversion factor is $\gamma(n)$.

2) The *a priori, a posteriori* estimation error and weight vector are computed as

$$\xi_M(n) = \varepsilon(n) / \gamma^{1/2}(n) \tag{15}$$

$$e_M(n) = \varepsilon(n)\gamma^{1/2}(n) \tag{16}$$

$$\hat{w}(n) = R^{-1}(n)p(n). \tag{17}$$

To annihilate the vector $u^T(n)$ in (14), $Q(n)$ is computed as a product of a sequence of Givens rotations, specifically, we write

$$Q(n) = \prod_{k=1}^{m} G_{m-k+1,n}(m-k+1) \tag{18a}$$

$$G_{i,j}(k) = \begin{bmatrix} I & & & & \mathrm{O} \\ & c & & s & \\ & & I & & \\ & -s & & c & \\ \mathrm{O} & & & & I \end{bmatrix} \begin{matrix} \\ \leftarrow \ i\text{th row} \\ \\ \leftarrow \ j\text{th row} \\ \\ \end{matrix} \tag{18b}$$

$$\begin{matrix} & \uparrow & & \uparrow & \\ \text{col. } i\text{th} & & j\text{th} & & \end{matrix}$$

where $i, j$ denotes the rows that will be affected when the Givens rotation matrix is premultiplied to $R(n\text{-}1)$ while $k$ gives the fact that input data in column k of $u^T(n)$ will be annihilated, and $c$ and $s$ are computed as

$$c = \frac{\left|R_{k,k}(n-1)\right|}{\sqrt{R_{k,k}^2(n-1) + u^2(n-k+1)}} \tag{19a}$$

$$s = \frac{u(n-k+1)}{R_{k,k}(n-1)}c. \tag{19b}$$

To pipeline the QRD-RLS systolic array at fine-grain level, a block updating scheme of the recursive computation is formulated with block size $M$ [14]-[15]. Specifically, input data vectors from clock cycle $n\text{-}M+1$ to $n$ are used to estimate the desired response $d(n)$. In order to achieve an exact exponential weighted LS solution, the forgetting factor $\lambda$ is included in the formulation:

$$\begin{bmatrix} R(n) & p(n) & R^{-1}(n)u(n) \\ \mathrm{O}_M^T & \varepsilon_M(n) & \Gamma^{1/2}(n) \end{bmatrix} = Q_M(n)\tilde{\Lambda}^{1/2}(M+1)\begin{bmatrix} R(n-M) & p(n-M) & 0 \\ U_M^T(n) & d_M(n) & \Delta \end{bmatrix} \tag{20}$$

where the $Q_M(n)$ is computed as

$$Q_M(n) = Q(n)Q(n-1)\cdots Q(n-M+1) \tag{21a}$$

where $Q_M(n)$ follows the definition of (18). We also define

$$\tilde{\Lambda}(M+1) = diag[\lambda^{M/2}I_m, \Lambda(M)] \tag{21b}$$

$$d_M^T(n) = [d(n-M+1)\cdots d(n-1)\ d(n)] \tag{21c}$$

$$U_M(n) = [u(n-M+1)\cdots u(n-1)u(n)] \tag{21d}$$

$$\Delta_M^T = [0,\cdots,1]. \tag{21e}$$

It is worth mentioning here that no matter how we choose the order of annihilation, $e(n)$ will always stay the same and is equal to $e(n)$ computed when the filter is not pipelined [15]. It is important that we rewrite the orthogonal matrix $Q_M(n)$ is

$$Q_M(n) = \begin{bmatrix} \mathcal{A}(n) & \mathrm{O} & \beta_2(n) \\ \mathrm{O} & I & \mathrm{O} \\ -\beta_1^T(n) & \mathrm{O} & \beta_3(n) \end{bmatrix} \tag{22}$$

where $\mathcal{A}(n)$ is an $m \times m$ matrix, $\beta_1(n)$ and $\beta_2(n)$ are both $m \times M$ matrices, and $\beta_3(n)$ is an $M \times M$ square matrix. Now we need to preserve the following three lemmas and theorems.

**Lemma 1** [15]: Given an upper triangular matrix $R(n\text{-}M)$ with positive diagonal elements and $U_M^T(n)$, as defined in (20), the corresponding $\mathcal{A}(n)$ and $\beta_2(n)$ defined in (22) are constant matrices.

**Lemma 2** [15]: Given an upper triangular matrix $R(n\text{-}M)$ with positive diagonal elements and $U_M^T(n)$, as defined in (20), the corresponding $\beta_3(n)$ defined in (22) is nonsingular.

**Lemma 3** [15]: Given an upper triangular matrix $R(n\text{-}M)$ with positive diagonal elements and $U_M^T(n)$, as defined in (20), the product $\beta_3^T\beta_3(n)$ is a constant matrix.

**Theorem 1** [15]: The system in (11)-(12) can

provide exact LS *a posteriori* estimation error vector, defined in (23b), which is computed as

$$e_M(n) = \beta_3^T(n)\varepsilon_M(n) \tag{23a}$$

where $e_M(n)$ is defined as

$$e_M(n) = d_M(n) - U_M^T(n)R^{-1}(n)p(n). \tag{23b}$$

**Theorem 2** [15]: The system in (11)-(12) can provide exact LS *a priori* estimation error vector, defined in (24b), which is computed as

$$\xi_M(n) = \beta_3^{-1}(n)\varepsilon_M(n) \tag{24a}$$

$$\xi_M(n) = d_M(n) - U_M^T(n-M)p(n-M). \tag{24b}$$

**Theorem 3** [15]: With an additional postprocessor added to the standard triangular array processor, the system in (11)-(12) is able to provide exact weight vector and a posteriori estimation error simultaneously (if $d(n)$ is not in the column space of $\mathcal{A}(n)$).

Applying the system in (10)-(12) to Kalman square-root information filters [8] is straight-forward since there is a one-to-one mapping between QRD-RLS variables and state-space variables. A signal flow graph (SFG) for updating an element in matrix $R(n\text{-}3)$ and a complete, 3-level pipelined QRD-RLS filter architectures can be found in [14]. The systolic array implementation is shown in Fig. 2(a). The non-recursive part of the systolic array structure can be pipelined easily. However, the speed (or sample rate) is still limited by the local recursive equation to update the content of the cell as

$$r(n) = c(n)\lambda^{1/2}r(n-1) + s(n)u_{in}(n) \tag{26a}$$

where *r(n)* is the cell content in the systolic array architecture, while *c(n) and s(n)* are the cosine and sine parameters of the Givens rotation, respectively. Applying $L$ levels of the look-ahead pipelining [6], we obtain

$$r(n) = \lambda^{L/2}r(n-L)\prod_{i=0}^{L-1}c(n-i) + \sum_{i=0}^{L-1}s(n-i)u(n-i). \tag{26b}$$

Thus, applying look-ahead pipelining in the recursive loop will result in a large amount of hardware overhead. The large complexity arises because the recursive equation contains a time-varying coefficient, namely $\lambda^{1/2}c(n)$.

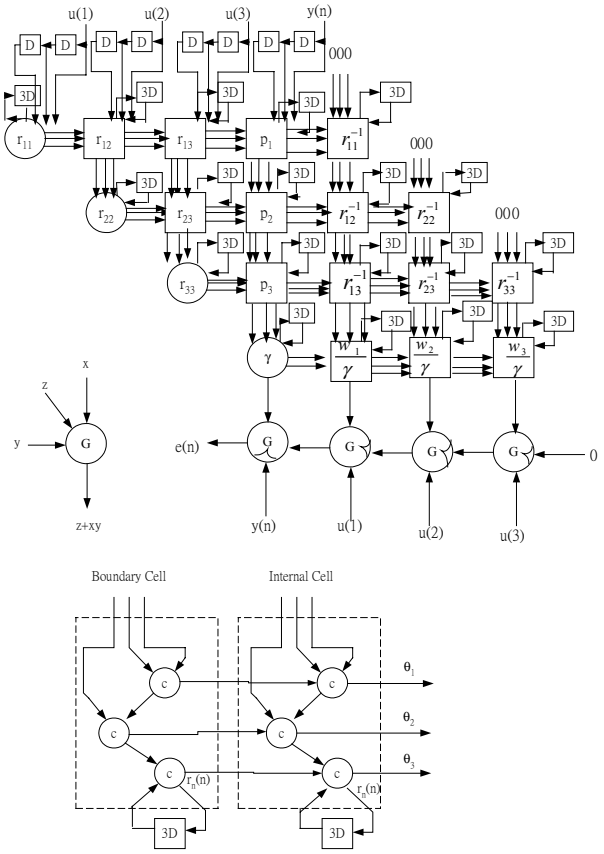
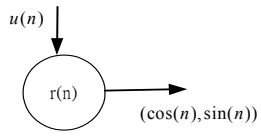



Fig. 2(a) Three-level fine-grain systolic implementation of the Pipeling Extended GR-RLS algorithm

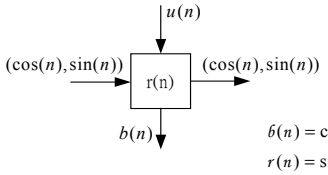
Fig. 2(b) Boundary Cell




$$r(n) = \sqrt{\lambda r(n-1)^2 + u(n)^2}$$

$$\cos(n) = \frac{\lambda^{1/2}r(n-1)}{r(n)}, \sin(n) = \frac{u(n)}{r(n)}$$

Fig. 2(c) Internal Cell




$$b(n) = \cos(n)u(n) - \sin(n)\lambda^{1/2}r(n-1) \quad (25a)$$

$$r(n) = \sin(n)u(n) + \cos(n)\lambda^{1/2}r(n-1) \quad (25b)$$

### *C. STAR-RLS Algorithm*

To overcome the problem of large hardware overhead in

pipelining the QRD-RLS algorithm, a scaled tangent rotation [2] has been proposed. The detail derivation of tangent of the STAR algorithm can be referred to [2]. Thus, we omit it here to save space.

### 3. PEGR-Recursive Least Square Filter (PEGR-RLS)

In this section, we will use a mathematical equation to approximate a square root equation, and a modified Givens Rotations Matrix is derived. We know that recursive equation within boundary cell within systolic array as implemented QRD-RLS algorithm is

$$r(n)=c(n)\lambda^{1/2}r(n-1)+s(n)x(n) \tag{27}$$

$$= \sqrt{\lambda r^2(n-1)+x^2(n)} \tag{28a}$$

$$= \lambda^{1/2}\left|r(n-1)\right|\left(1+\frac{x^2(n)}{\lambda r^2(n-1)}\right)^{1/2} \tag{28b}$$

$$= \left|x(n)\right|\left(1+\frac{\lambda r^2(n-1)}{x^2(n)}\right)^{1/2} \tag{28c}$$

Now, consider two cases. One is

$$\text{A):}\ \ \left|u_{in}(n)\right| \geq \lambda^{1/2}\left|r(n-1)\right|;$$

and the other is

$$\text{B):}\ \left|u_{in}(n)\right| < \lambda^{1/2}\left|r(n-1)\right|.$$

(I) Case A):

Because of case A, we can express it as

$$u_{in}^2(n) \geq \lambda r^2(n-1);\ \Rightarrow\ \left|\frac{\lambda r^2(n-1)}{x^2(n)}\right| < 1.$$

Then, we use Taylor series expansion for the function [13]

$$a(1+x)^p = a\left(1+px+\frac{p(p-1)}{2!}x^2+\cdots\right),\quad for\ |x|<1. \tag{29}$$

Neglecting the higher order terms, (28c) be rewritten as

$$r(n) = \lambda^{1/2}r(n-1)\left(1+\frac{1}{2}\times\frac{u_{in}^2(n)}{\lambda r^2(n-1)}\right) \tag{30}$$

$$\cong \lambda^{1/2}\left|r(n-1)\right|+\frac{1}{2}\times\frac{u_{in}^2(n)}{\lambda^{1/2}\left|r(n-1)\right|}.$$

When PEGR-RLS algorithm reaches a steady state, we can find something from (28a) that *r(n)* and *r(n-1)* have the same positive sign. So (30) can be got

$$r(n)\cong\lambda^{1/2}r(n-1)+\frac{1}{2}\times\frac{u_{in}^2(n)}{\lambda^{1/2}r(n-1)}. \tag{31}$$

(31) is compared with (25), and we can find new modified $c_{11}(n)$ and $s_{12}(n)$, respectively as

$$c_{11}(n)=1 \tag{32a}$$

and

$$s_{12}=\frac{x(n)}{2\lambda^{1/2}r(n-1)}. \tag{32b}$$

Then, $c_{22}(n)$ and $s_{21}(n)$ also can be approximated as

$$c_{22}(n)=\frac{\lambda^{1/2}r(n-1)}{r(n)}=\frac{\lambda^{1/2}r(n-1)}{\lambda^{1/2}r(n-1)+\frac{1}{2}\times\frac{x^2(n)}{\lambda^{1/2}r(n-1)}}\cong 1 \tag{32c}$$

$$s_{21}(n)=\frac{x(n)}{r(n)}\cong\frac{x(n)}{\lambda^{1/2}r(n-1)+\frac{1}{2}\times\frac{x^2(n)}{\lambda^{1/2}r(n-1)}}\cong\frac{x(n)}{\lambda^{1/2}r(n-1)}. \tag{32d}$$

Accordingly, from (32a) to (32d), we can get the first extended Givens rotations as

$$\begin{bmatrix} 1 & \frac{x(n)}{2\lambda^{1/2}r(n-1)} \\ -\frac{x(n)}{\lambda^{1/2}r(n-1)} & 1 \end{bmatrix}.$$

(II) Case B): $\left|u_{in}(n)\right| < \lambda^{1/2}\left|r(n-1)\right|$

Because case B, we can get an equation as

$$\lambda\, r^2(n-1) < u_{in}^2(n).$$

After using use Taylor series expansion [13], we can approximate (28c) as

$$r(n)=\left|x(n)\right|\left(1+\frac{\lambda r^2(n-1)}{x^2(n)}\right)^{1/2}\cong\left|x(n)\right|+\frac{\lambda r^2(n-1)}{2\left|x(n)\right|}. \tag{33}$$

When PEGR-RLS algorithm reaches a steady state, we can get

$$\left|r(n)\right| = \left|r(n-1)\right| = \alpha_1\left|x(n)\right| \tag{34a}$$

where

$$1 < \alpha_1 < \sqrt{2}. \tag{34b}$$

It is the reason that we want to get a pipelined EGR-RLS algorithm, so we let $\alpha_1 \cong 1$ in (34). From (28c), we can

find that *r(n)* and *r(n-1)* have the same sign. Thus, (33) can be rewritten as $r(n) \cong |x(n)| + \frac{r(n-1)}{2}$. Compared with (34), we can obtain the second relaxed *c(n)* and *s(n)*, respectively

$$c_{11}(n) = \frac{\lambda^{1/2}}{2} \tag{35a}$$

$$s_{12}(n) = sign(x(n)) \tag{35b}$$

$$c_{22}(n) = \frac{\lambda^{1/2} r(n-1)}{r(n)} \cong \frac{\lambda^{1/2} r(n-1)}{|x(n)|} \tag{35c}$$

$$s_{12}(n) = \frac{x(n)}{r(n)} \cong \frac{x(n)}{|x(n)|} = sign(x(n)). \tag{35d}$$

From (35a) to (35d), we get the second extended Givens rotations as

$$\begin{bmatrix} \frac{\lambda^{1/2}}{2} & sign\ (x(n)) \\ -\ sign\ (x(n)) & \frac{\lambda^{1/2} r(n-1)}{|x(n)|} \end{bmatrix}. \tag{36}$$

Before we discuss the maximum approximation error, we first analyze *r(n)* and *x(n)* when they are reached steady state. We modified the (35a) as

$$r^2(n) = \lambda r^2(n-1) + x^2(n) \tag{37}$$

When the algorithm is reached steady state, we can assume *r(n) = r(n-1)*. Thus, we can obtain

$$\frac{x^2(n)}{r^2(n)} = 1 - \lambda. \tag{38}$$

The maximum approximation error is occurred when

$$\lambda^{1/2} |r(n-1)| = |x(n)| \tag{39}$$

and using (37)-(39), we find that $\lambda$ = 1/2. But the simulation result is shown that extended Givens rotation RLS algorithm is close to QRD-RLS algorithm when $\lambda$ = 1/2. The function of processing element within systolic array is shown in Fig 3(a). The boundary cell and internal cell, which are used, relaxed Givens rotations block diagram is shown in Fig 4 and Fig 5, respectively.

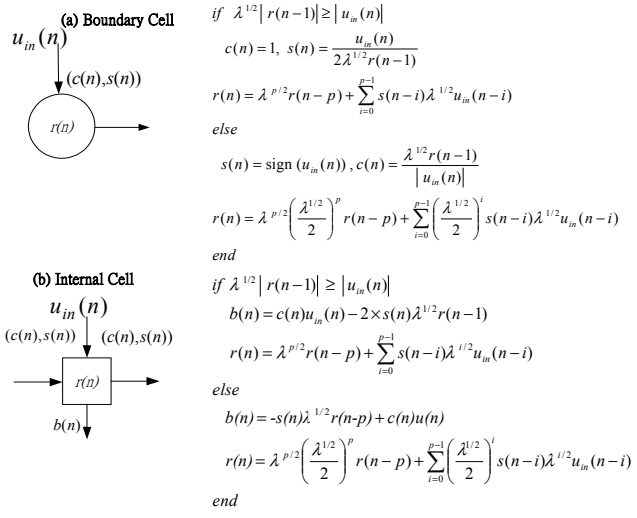



Fig. 3. Boundary cell and Internal cell for PEGR-RLS

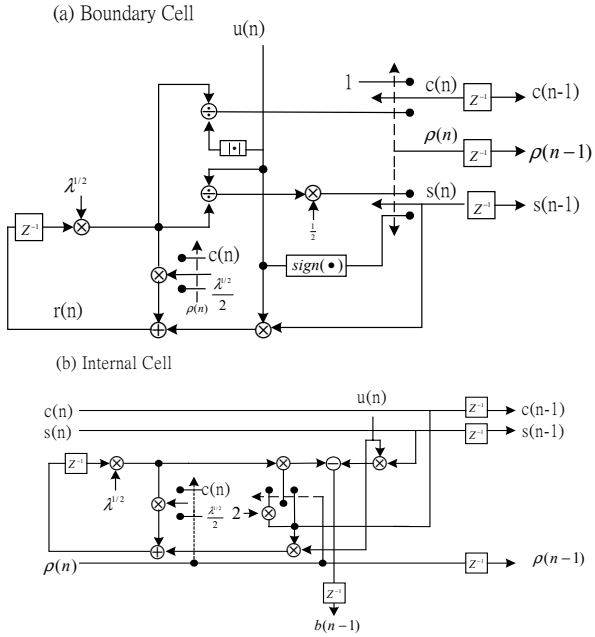



Fig.4. The architectures of Boundary cell and Internal cell for non-pipelined EGR-RLS.

## 4. Simulation Results

The simulation example has been used in [4] and it was conducted using the 11-tap equalizer. Its block diagram is shown in Fig 1. A random binary sequence *d(n)*, which takes values +1 or –1, is used as the input to the channel. The channel has an impulse response of a raised cosine

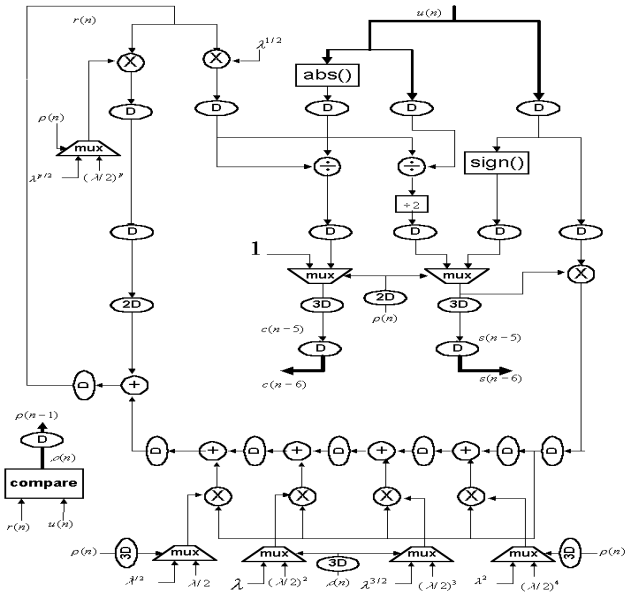



Fig. 5(a). The architecture of Boundary Cell for PEGR-RLS with pipeline level (p = 5).

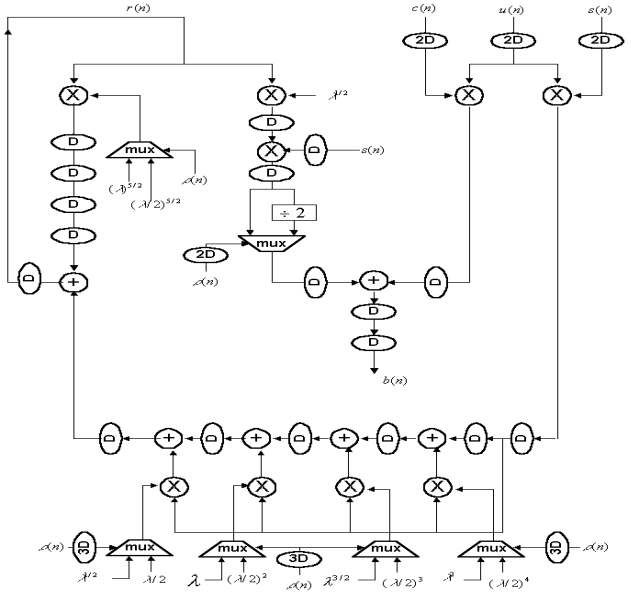



Fig. 5(b). The architecture of Internal Cell for PEGR-RLS with pipeline level (p = 5).

$$h_n = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{W}(n-2)\right)\right], n=1,2,3 \\ 0, \quad otherwise \end{cases}$$

where $W$ is the amplitude of noise, and is chosen to be 3.3. The output of channel $u(n)$ is given by

$$u(n) = \sum_{i=1}^{3} h_i d(n-i) + x(n),$$

where $x(n)$ is the additive noise, which is modeled to be a Gaussian zero-mean random variable with a variance $\sigma_x^2 = 0.001$. The signal $d(n)$ is used as the desired signal for the algorithm. In Fig 6, we can compare the convergence of the mean squared error for the STAR-RLS, QRD-RLS, and PSTAR-RLS algorithms.

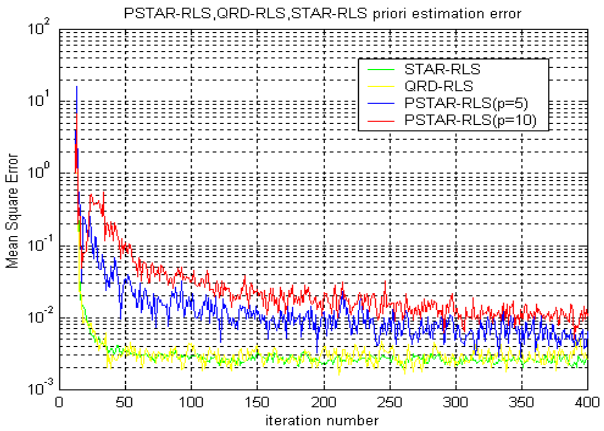



Fig 6. Simulation results for QRD-RLS, STAR-RLS, and PSTAR-RLS algorithm.

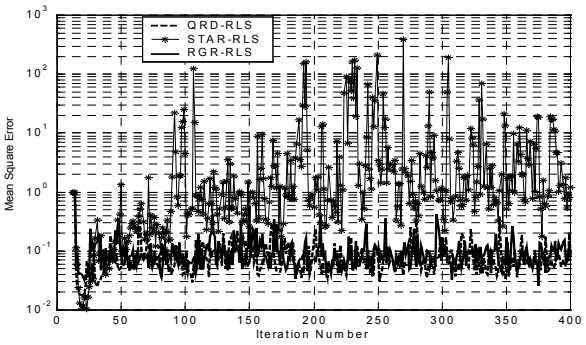



Fig. 7. Simulation results of QRD-RLS, STAR-RLS, and EGR-RLS algorithm for 3 different forgetting factors with $\lambda$ = 0.02, 0.5and 0.98, respectively.

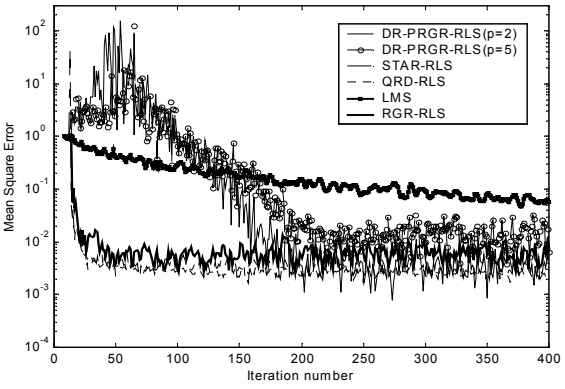


Fig. 8. Simulation results of QRD-RLS, STAR-RLS, and PEGR-RLS algorithms for 3 different forgetting factors with $\lambda$ = 0.02, 0.5 and 0.98, for p = 2 & 5, respectively.

## 5. Conclusions

The plots for the PSTAR-RLS algorithm are shown for speedups of 5 and 10 times. There is a gradual degradation in convergence rate with higher speedup, as would be expected. However, at all speedups, the steady-state error does converge finally to that of the serial algorithm. The retiming technique is the method we adopt to change the locations of delay elements in a circuit without affecting the input/output characteristics of the circuit. The lookahead, delay relaxation, and retiming methods are the core pipelining technology to implement the PEGR-RLS. Specifically, it can be pipelined at very low forgetting factor by using extended look-ahead. Simulation results are presented to compare the performance of the STAR-RLS, QRD-RLS, and LMS algorithms.

## References


[1] Kung, H. T. and C. E. Leiserson, "Systolic arrays (for VLSI)," *Sparse Matrix Proc.* 1978, *Soc. Ind. Appl. Math*., 1978, pp. 256-182. [A version of this paper is reproduced in Mead and Conway, 1980].

[2] Raghunath, K.J.; Parhi, K.K. "Pipelined RLS adaptive filtering using scaled tangent rotations (STAR)" *Signal Processing, IEEE Transactions on* Vol. 44 10 , Oct. 1996 , Page(s): 2591 –2604.

[3] L.D. Van, "Design of efficient VLSI architectures: multipliers, 2-D digital filter, and adaptive digital filter," *Ph.D. dissertation*, Dept. Elect. Eng., National Taiwan University, Taipei, Taiwan, R.O.C., 2001.

[4] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall ,1986

[5] Keshab K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, John Wiley & Sons, Inc. 1999

[6] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE, Trans. Acoust., Speech, Signal Prcossing*, vol. 37, pp. 1118-1134, July 1989.

[7]. K. K. Parhi, and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part II: Pipelining incremental block filtering," *IEEE, Trans. Acoust., Speech, Signal Prcossing*, vol. 37, pp. 1099-1117, July 1989.

[8] T. H. Y. Meng, E. A. Lee, and D. G. Messerschmitt, "Least-squares computation at arbitrarily high speeds," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Processing* (ICASSP), 1987, pp. 1398-1401.

[9] A. H. Sayed, T. Kailath, "A State-Space Approach to Adaptive RLS Filtering," *IEEE Trans. Signal Processing Magazine*, vol. 42, pp. 18-60, Jul. 1994.

[10] S. F. Hsieh, K. J. R. Liu, and K. Yao, "A unified square-root-free Givens rotations approach for QRD-based recursive least squares estimation," *IEEE Trans. Signal Processing*, vol. 41, pp. 1405-1409, Mar. 1993.

[11] W. M. Gentleman, "Least-squares computations by Givens transformations without square-roots," *J. Inst. Math. Applicat*., vol. 12, pp.329-336, 1973.

[12] F. Ling, "Givens rotation based least-squares lattice and related algorithm," *IEEE Trans. Signal Processing*, vol. 39, pp. 1541-1551, Jul. 1991.

[13] G. J. Borse, Numerical Methods with MATLAB, A Resource for Scientists and Engineers, PWS Publishing Company 1997.

[14] J. Ma, K. K. Parhi, and E. F. Deprettere, "Annihilation-reordering look-ahead pipelined CORDIC based RLS adaptive filters and their application to adaptive beamforming," *IEEE Trans. Signal Processing*, vol. 48, pp. 2414-2431, Aug. 2000.

[15] Z. Chi, J. Ma, K. K. Parhi, "Hybrid Annihilation Transformation (HAT) for Pipelining QRD-Based Least-Square Adaptive Filters", *IEEE Trans. On Circuits and Systems –II, Analog and Digital Signal Processing*, vol. 48, No. 7, pp. 661-674, Jul. 2001.