# TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER

*Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry*

# A Unified Approach for QRD-Based Recursive Least-Squares Estimation without Square Roots

*by S.F. Hsieh, K.J.R. Liu and K. Yao*

TR 91-71

# A Unified Approach For QRD-based Recursive Least-Squares Estimation Without Square Roots

**S.F. Hsieh**
Dept. of Communication Engineering
Nat'l Chiao Tung University
Hsinchu, Taiwan 30039

**K.J.R. Liu**
Electrical Engineering Dept.
Systems Research Center
University of Maryland
College Park, MD 20742

**K. Yao**
Electrical Engineering Dept.
UCLA
Los Angeles, CA 90024-1594

## ABSTRACT

The QR-decomposition (QRD)-based recursive least-squares (RLS) methods have been shown to be useful and effective towards *adaptive signal processing* in modern communications, radar, and sonar systems implementable with various modern parallel and systolic array architectures. The planar (Givens) and hyperbolic rotations are the most commonly used methods in performing the QRD up/downdating. But the generic formula for these rotations require explicit square-root (sqrt) computations, which constitute the computational bottleneck and are quite undesirable from the practical VLSI circuit design point of view. There has been more than ten sqrt-free algorithms known so far. In this paper, we provide a unified systematic approach for the sqrt-free QRD-based RLS estimation problem. By properly choosing two parameters, $\mu$ and $\nu$, all existing known sqrt-free methods fall in the category of our unified approach. The proposed method not only can generalize all currently known sqrt-free QRD algorithms, but also new sqrt-free algorithms as long as the parameters $\mu$ and $\nu$ are properly chosen. The unified treatment is also extended to the QRD-based RLS problems for optimal residual acquisition without sqrt operations, and the systolic array implementation.

# 1 Introduction

The Givens rotation is a most versatile method in performing many signal processing algorithms involving matrix computations such as the QR decomposition, the singular value decomposition, and the eigen-decomposition [6]. Adaptive antenna arrays, beamforming, superresolution frequency estimation, system identification, and other real-time applications make immediate the quest for fast algorithms and parallel architectures. While many researchers have worked on reformulating algorithms suitable for parallel computing and VLSI architectures, the current VLSI architectures still disfavor, if not prohibit, sophiscated computations. A noticeable example is the *square root* (sqrt) operation which not only occupies much area in a VLSI chip but also requires longer time to accomplish such computation. Numerical analysis also shows that the accuracy of a sqrt operation plays an important role under finite precision computation, which discourages any shortcut in roughly performing the sqrt operation. As a simulation study resented in [12], Proudler *et al.* showed that finite-precision implementation of the sqrt-free algorithms did have better numerical results than that of the conventional Givens rotation method.

With this in mind, many efforts have been spent on how to minimize, or even eliminate, the sqrt operations, from the algorithms. One famous example is the sqrt-free Givens rotation first proposed by Gentleman [4]. Hammarling generalized his results shortly [8]. Later on other versions of the sqrt-free Givens rotations were also proposed [1, 2, 7]. All of the above algorithms only focus on sqrt-free Givens itself and/or its

applications in solving a least squares (LS) problem. McWhirter [11] is the first one who applied the sqrt-free Givens rotation to recursive least squares problems in computing the desired optimum residual without solving for the least-squares coefficients. Closely related to the Givens rotation method, the modified Gram-Schmidt (MGS) orthogonation is another approach in performing the QR decomposition. Ling, *et al.* [10] and Kalson and Yao [9] independenly developed the sqrt-free MGS methods for the recursive LS (RLS) filtering problems. Recently, Chen and Yao [3] summarized the works done on the sqrt-free RLS filtering and proposed another more efficient sqrt-free method. In fact, there has been more than ten sqrt-free algorithms known so far [4, 8, 1, 9, 10, 2, 3, 7]. All of the approaches are heuristic. There is no systematic way of generating the sqrt-free algorithms. Motivated by these works, we try to ask and answer the following question: *what is the crux embedded in the sqrt-free problems?* In another words, is there any fundamental relationships among these sqrt-free algorithms? The answer is yes! One of the major contributions in this paper is that we did find this fundamental relationship and are able to describe it by using only two parameters.

In Section 2, planar (or Givens) rotation and its simili, hyperbolic rotation, are introduced. The prototypes of generalized sqrt-free algorithms follow in Section 3, where we found surprisingly all of those proposed sqrt-free algorithms can be found by proper choice of two parameters. Another noticeable observation is the relationship between Givens rotation and rank-1 MGS updating. We proceed in Section 4 with seeking a sqrt-free optimum residual as required in RLS filtering problems. Section 5

compares the systolic implementations and conclusions are given in Section 6.

## 2 Planar and Hyperbolic Rotations

There are two basic operations in the QR-based algorithms, namely updating and downdating. The updating operation is to update current information when a new data arrives, while the downdating is to remove old data if certain undesired information needs to be dropped. A $2 \times 2$ *planar* (Givens) rotation matrix is the most fundamental orthogonal matrix in performing the updating of the QR decomposition. A *planar* (Givens) rotation is given by

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

and is used to premultiply a two-row matrix

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \\ \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix}$$

to zero the element in the $(2,1)$ location such that it becomes

$$\begin{bmatrix} \alpha'_1 & \alpha'_2 & \cdots & \alpha'_p \\ 0 & \beta'_2 & \cdots & \beta'_p \end{bmatrix},$$

where

$$c = \alpha_1 / \sqrt{\alpha_1^2 + \beta_1^2}\,, \tag{1}$$

$$s = \beta_1 / \sqrt{\alpha_1^2 + \beta_1^2}\,, \tag{2}$$

$$\alpha_1' = \sqrt{\alpha_1^2 + \beta_1^2}\,, \tag{3}$$

$$\alpha_j' = c\alpha_j + s\beta_j\,, \tag{4}$$

$$j = 2, \cdots, p\,.$$

$$\beta_j' = -s\alpha_j + c\beta_j\,, \tag{5}$$

The information contained in $\beta_1$ has now been absorbed and updated in $\alpha_i', i = 1, \cdots, p$. This planar rotation scheme can be applied to zero the following $\beta_i', i = 2, \cdots, p$, so that the information contained in the row of $\beta_i$ will eventuately be updated into the row of $\alpha_i$.

Similarly, a $2 \times 2$ *pyperbolic* rotation matrix is the most fundamental orthogonal matrix in performing the downdating of the QR decomposition. A *hyperbolic* rotation matrix is given by

$$\begin{bmatrix} \hat{c} & -\hat{s} \\ -\hat{s} & \hat{c} \end{bmatrix},$$

and the corresponding parameters satisfy:

$$\hat{c} = \alpha_1/\sqrt{\alpha_1^2 - \beta_1^2}\,, \tag{6}$$

$$\hat{s} = \beta_1/\sqrt{\alpha_1^2 - \beta_1^2}\,, \tag{7}$$

$$\alpha_1' = \sqrt{\alpha_1^2 - \beta_1^2}\,, \tag{8}$$

$$\alpha_j' = \hat{c}\alpha_j - \hat{s}\beta_j\,, \tag{9}$$

$$j = 2, \cdots, p\,.$$

$$\beta_j' = -\hat{s}\alpha_j + \hat{c}\beta_j\,, \tag{10}$$

Different from the updating, in the downdating, the information about the row of $\beta_i$ is removed from the row of $\alpha_i$ through the hyperbolic rotation. Both rotations provide

us the basic operations to add or delete information that is necessary or undesired.

# 3 The $\mu\nu$ Family of Square-Root-Free Algorithms

In VLSI circuit design, square-root operation is expensive, because it takes up much area or is slow (due to many iterations). Therefore, it is advantageous to avoid square-root operations or minimize the required number of such operations. We will focus on how to meet this goal for the $2 \times 2$ planar and hyperbolic rotations.

By taking out a scaling factor from each row, the two rows under consideration before and after the orthogonal transformations are given by

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \\ \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix} = \begin{bmatrix} \sqrt{k_a} & 0 \\ 0 & \sqrt{k_b} \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_p \\ b_1 & b_2 & \cdots & b_p \end{bmatrix} \tag{11}$$

and

$$\begin{bmatrix} \alpha'_1 & \alpha'_2 & \cdots & \alpha'_p \\ 0 & \beta'_2 & \cdots & \beta'_p \end{bmatrix} = \begin{bmatrix} \sqrt{k'_a} & 0 \\ 0 & \sqrt{k'_b} \end{bmatrix} \begin{bmatrix} a'_1 & a'_2 & \cdots & a'_p \\ 0 & b'_2 & \cdots & b'_p \end{bmatrix}, \tag{12}$$

where $k_a, k_b, k'_a$, and $k'_b$ are the scaling factors for sqrt-free operations, and $\alpha'_i$ and $\beta'_i$ are the updated $\alpha_i$ and $\beta_i$ when the $\beta_1$ is zeroed out.

Now, our task is to find the expressions for $k'_a, k'_b, a'_1$, $\{(a'_j, b'_j),\ j = 2, \cdots, p\}$, in terms of $k_a, k_b, \{(a_j, b_j),\ j = 1, \cdots, p\}$, such that *NO* square-root operation is actually needed. The square-root expressions of $\sqrt{k_a}$, $\sqrt{k_b}$, $\sqrt{k'_a}$, and $\sqrt{k'_b}$ in (11) and (12) are used for representational purposes only and are not actually performed.

For simplicity, let us focus on planar rotations only, similar derivations for the square-root-free hyperbolic rotation can also be obtained by replacing $k_b$ with $-k_b$,

and $k'_b$ with $-k'_b$. Replacing $\alpha_j = \sqrt{k_a}a_j$, $\beta_j = \sqrt{k_b}b_j$, $\alpha'_j = \sqrt{k'_a}a'_j$, $\beta'_j = \sqrt{k_b}b'_j$, $j = 1, \ldots, p$, in (1) - (5) leads to

$$c = \sqrt{k_a}\, a_1 \,/\, \sqrt{k_a a_1^2 + k_b b_1^2}\,, \tag{13}$$

$$s = \sqrt{k_b}\, b_1 \,/\, \sqrt{k_a a_1^2 + k_b b_1^2}\,, \tag{14}$$

$$\sqrt{k'_a}\, a'_1 = \sqrt{k_a a_1^2 + k_b b_1^2}\,, \tag{15}$$

$$\sqrt{k'_a}\, a'_j = \frac{k_a\, a_1}{\sqrt{k_a a_1^2 + k_b b_1^2}}\, a_j + \frac{k_b\, b_1}{\sqrt{k_a a_1^2 + k_b b_1^2}}\, b_j\,, \tag{16}$$

$$j = 2, \cdots, p.$$

$$\sqrt{k'_b}\, b'_j = \frac{-\sqrt{k_a k_b}\, b_1}{\sqrt{k_a a_1^2 + k_b b_1^2}}\, a_j + \frac{\sqrt{k_a k_b}\, a_1}{\sqrt{k_a a_1^2 + k_b b_1^2}}\, b_j\,, \tag{17}$$

After simplifications we have

$$a'_1 = \sqrt{\frac{k_a a_1^2 + k_b b_1^2}{k'_a}}\,, \tag{18}$$

$$a'_j = \frac{1}{\sqrt{k'_a}\,\sqrt{k_a a_1^2 + k_b b_1^2}}[k_a a_1 a_j + k_b b_1 b_j]\,, \tag{19}$$

$$j = 2, \cdots, p.$$

$$b'_j = \frac{\sqrt{k_a k_b}}{\sqrt{k'_b}\,\sqrt{k_a a_1^2 + k_b b_1^2}}[-b_1 a_j + a_1 b_j]\,, \tag{20}$$

To avoid square-roots computation, we need to determine $k'_a$ and $k'_b$ such that $a'_1$, $a'_j$ and $b'_j$ will not require square-root operations. It is clear that if we choose $k'_a$ and $k'_b$ as

$$k'_a = \frac{k_a a_1^2 + k_b b_1^2}{\mu^2}\,, \tag{21}$$

$$k'_b = \frac{k_a k_b}{\nu^2(k_a a_1^2 + k_b b_1^2)}\,, \tag{22}$$

where $\mu$ and $\nu$ are parameters that will be determined later to be any square-root-free functions of $k_a, k_b, a_1$, and $b_1$, then (18) - (20) can be computed without square-roots

operations. We thus have the following updating formulas without square-roots:

$$k'_a = \frac{k_a a_1^2 + k_b b_1^2}{\mu^2} , \tag{23}$$

$$k'_b = \frac{k_a k_b}{\nu^2 (k_a a_1^2 + k_b b_1^2)} = \frac{k_a k_b}{\mu^2 \nu^2 k'_a} , \tag{24}$$

$$a'_1 = \mu , \tag{25}$$

$$\begin{aligned} a'_j &= \frac{\mu}{k_a a_1^2 + k_b b_1^2} [k_a a_1 a_j + k_b b_1 b_j] \\ &= \frac{k_a a_1 a_j + k_b b_1 b_j}{\mu k'_a} , \end{aligned} \tag{26}$$

$$j = 2, \cdots, p ,$$

$$b'_j = \nu[-b_1 a_j + a_1 b_j] , \tag{27}$$

$$c = \frac{a_1}{\mu} \sqrt{\frac{k_a}{k'_a}} , \tag{28}$$

$$s = \frac{b_1}{\mu} \sqrt{\frac{k_b}{k'_a}} . \tag{29}$$

Notice that the square-root operations disappear in our formulas of (23) - (27) needed in the planar and hyperbolic rotations. Also, the $c$ and $s$ in (28) and (29) are not explicitly necessary in the computation of (23) - (27). The use of the rotation parameter $c$ in (28) (with a square-root operation) will be further considered in the next section when the optimum residual $e$ is desired. Next Section will show that it is possible to obtain $e$ without any square-root operation where the explicit computation of the rotation parameter $c$ can be bypassed. To avoid repetitive computations and take the advantage of previous computed results, (24), (26), (28) and (29) use the newly updated $k'_a$ of (23). As stated earlier, we are free to choose those two parameters $\mu$ and $\nu$. Different choices of $\mu$ and $\nu$ will affect the number of multiplications and divisions, as well as the numerical stability and parallelism of computations.

It can be easily shown that this unified view can generate *all* of the previously known researches on the square-root-free algorithms via an proper choice of $\mu$ and $\nu$. In fact, there has been more than ten sqrt-free algorithms known so far. Among them are Gentleman [4], Hammarling [8], Bareiss [1], Kalson and Yao [9], Ling, *etc.* [10], Barlow and Ispen [2], Chen and Yao [3], Götze and Schwiegelshohn [7]. For example, if we choose $\mu = 1$ and $\nu = 1$, it becomes the sqrt-free algorithm proposed by Gentleman in [4] and can be updated as follows,

$$\begin{aligned} k'_a &= k_a a_1^2 + k_b b_1^2 \\ k'_b &= \frac{k_a k_b}{k'_a} , \\ a'_1 &= 1 , \qquad (30) \\ a'_j &= \frac{k_a a_1 a_j + k_b b_1 b_j}{k'_a} , \qquad j = 2, \cdots, p , \\ b'_j &= -b_1 a_j + a_1 b_j . \qquad (31) \end{aligned}$$

(32)

To check if this is correct, we find that

$$\alpha'_j = \sqrt{k'_a} a'_j = \frac{k_a a_1 a_j + k_b b_1 b_j}{\sqrt{k'_a}} = \frac{\alpha_1 \alpha_j + \beta_1 \beta_j}{\sqrt{\alpha_1^2 + \beta_1^2}} = c_1 \alpha_j + s_1 \beta_j, \qquad (33)$$

and

$$\beta'_j = \sqrt{k'_b} b'_j = \frac{\sqrt{k_a k_b} b'_j}{\sqrt{k'_a}} = \frac{\alpha_1 \beta_j - \beta_1 \alpha_j}{\sqrt{\alpha_1^2 + \beta_1^2}} = -s\alpha_j + c\beta_j, \qquad (34)$$

which are consistent with the results in (4) and (5). For the systolic array implementation described in [11], specifically we choose $a_1 = 1$ and define the generalized

rotation parameters

$$\bar{c} = \frac{k_a}{k'_a}, \quad \bar{s} = \frac{k_b b_1}{k'_a}, \tag{35}$$

we have

$$k'_a = k_a + k_b b_1^2, \tag{36}$$

$$a'_j = \bar{c} a_j + \bar{s} b_j, \tag{37}$$

$$b'_j = b_j - b_1 a_j. \tag{38}$$

These results are consistent with the works by Gentleman [4] and McWhirter [11]. The details of the systolic implementation can be found in [11].

In Table 1, we list various square-root-free algorithms and the corresponding choices of $\mu$ and $\nu$. It is quite clear now that these sqrt-free algorithms are closely related by the parameters $\mu$ and $\nu$. Hence, this catagory of sqrt-free algorithms is called the $\mu\nu$ family of the sqrt-free algorithms.

Now, not only can we generate those known sqrt-free algorithms, but also are we able to find *new* sqrt-free algorithms by choosing new pairs of $(\mu, \nu)$ parameters. Of course, they will all belong to the $\mu\nu$ family. As an example, let us choose

$$\mu = k_a a_1^2 + k_b b_1^2, \tag{39}$$

$$\nu = \frac{1}{k_a a_1^2 + k_b b_1^2}, \tag{40}$$

one can readily verify that this is a new sqrt-free algorithm. In principle, there are unlimited choices of $\mu$ and $\nu$ for sqrt-free algorithms.

Table 1: Some known sqrt-free algorithms of the $\mu\nu$ family

| $\mu$ | $\nu$ | Authors(year) | Remark |
|---|---|---|---|
| 1 | 1 | Gentleman('73) | $a_1 = 1$ |
| $\frac{k_a a_1^2 + k_b b_1^2}{k_b b_1}$ | $\frac{-1}{b_1}$ | " | |
| $\frac{k_a a_1^2 + k_b b_1^2}{k_a a_1}$ | $\frac{1}{a_1}$ | " | |
| $\frac{k_a a_1^2 + k_b b_1^2}{k_a a_1}$ | $\frac{1}{a_1}$ | Hammarling('74) | |
| $\frac{k_a a_1^2 + k_b b_1^2}{k_a a_1}$ | $\frac{-1}{b_1}$ | " | |
| $\frac{k_a a_1^2 + k_b b_1^2}{k_a a_1}$ | $\frac{k_b b_1}{k_a a_1^2}$ | " | |
| 1 | $\frac{1}{a_1}$ | Bareiss('82) | |
| $a_1 + k_b b_1^2$ | $\frac{1}{a_1 + k_a b_1^2}$ | Ling('86),Kalson/Yao('85) | $k_a a_1 = 1$ |
| $\frac{k_a a_1^2 + k_b b_1^2}{k_a a_1}$ | $\frac{1}{a_1}$ | Chen/Yao ('88) | $k_a a_1 = 1$ |
| $\frac{k_a a_1^2 + k_b b_1^2}{k_a k_b}$ | 1 | Götze/Schwiegelshohn('89) | |
| $2^{\tau_1}(k_a a_1^2 + k_b b_1^2)$ | $2^{\tau_2}$ | Barlow/Ispen('87) | scaled |

# 4 Sqrt-Free Triangular Array Updating and Optimum Residual Acquisition

Solving a lesat-squares problem $\|A\mathbf{w} \approx \mathbf{b}\|$ without the square root operation can be easily achieved [4]. Let the QRD of $A$ be $A = QR$, where $R$ is an upper triangular matrix, then

$$Q^T[A\ \mathbf{b}] = \begin{bmatrix} R & \mathbf{u} \\ \mathbf{0} & \mathbf{v} \end{bmatrix},$$

and the optimal weight vector can be obtained by sovling

$$R\hat{\mathbf{w}} = \mathbf{u}.$$

Now, since a diagonal matrix $D^{\frac{1}{2}}$ has been factorized from $A$ so that $[A\ \mathbf{b}] = D^{\frac{1}{2}}[\bar{A}\ \bar{\mathbf{b}}]$. Let

$$\bar{Q}^T[\bar{A}\ \bar{\mathbf{b}}] = \begin{bmatrix} \bar{R} & \bar{\mathbf{u}} \\ \bar{\mathbf{0}} & \bar{\mathbf{v}} \end{bmatrix},$$

The optimal residual can be obtained without the square root operation by solving

$$\bar{R}\hat{\mathbf{w}} = \bar{\mathbf{u}}.$$

In this section, we will apply the prototypes of sqrt-free rotations developed above to the QRD-based recursive LS (RLS) estimation problems such as the sidelobe canceller adaptive beamforming [11] and adaptive filtering. In these applications, the weight vector is not explicitly necessary, instead, we are interested in the optimal residual. It has been well-known how to obtain the optimal residual by using the

systolic array [11]. To be specific, we are interested in updating from

$$\begin{bmatrix} R & \mathbf{u} \\ \mathbf{x}^T & y \end{bmatrix} \tag{41}$$

to

$$\begin{bmatrix} R' & \mathbf{u}' \\ \mathbf{0}^T & v \end{bmatrix}. \tag{42}$$

It has been shown [11] that the $p \times p$ upper triangular matrix $R'$ can be obtained through a sequence of $p$ Givens rotations and the optimum residual $e$ for the newly appended data $[\,\mathbf{x}^T \vdots y\,]$ is given by

$$e = -(\prod_{i=1}^{p} c_i)\, v, \tag{43}$$

with $c_i$ representing the cosine value of the $i$-th rotation angle.

Factoring out the scaling constants into the pre-multiplying diagonal matrix leads (41) to the form of

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} & u_1 \\ & r_{22} & \cdots & r_{2p} & u_2 \\ & & \ddots & \vdots & \vdots \\ & & & r_{pp} & u_p \\ x_1 & x_2 & \cdots & x_p & y \end{bmatrix} \tag{44}$$

$$
= \begin{bmatrix} \sqrt{k_1} & & & & \\ & \sqrt{k_2} & & & \\ & & \ddots & & \\ & & & \sqrt{k_p} & \\ & & & & \sqrt{k_q} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} & a_{1,p+1} \\ & a_{22} & \cdots & a_{2p} & a_{2,p+1} \\ & & \ddots & \vdots & \vdots \\ & & & a_{pp} & a_{p,p+1} \\ b_1 & b_2 & \cdots & b_p & b_{p+1} \end{bmatrix} . \tag{45}
$$

Unlike the previously developed formula, where we are only interested in updating $k_i, a_{ij}$, to $k'_i, a'_{ij}$ and zeroing out all the $b_i$'s, this time we do also need to know the cosine values explicitly as required in the optimum residual given in (43).

After the first rotation, $b_1$ will be zeroed-out and we have

$$
k'_1 = \frac{k_1 a_{11}^2 + k_q b_1^2}{\mu_1^2} , \tag{46}
$$

$$
k_q^{(1)} = \frac{k_1 k_q}{\mu_1^2 \nu_1^2 k'_1} , \tag{47}
$$

$$
a'_{11} = \mu_1 , \tag{48}
$$

$$
a'_{1j} = \frac{1}{\mu_1 k'_1} \left[ k_1 a_{11} a_{1j} + k_q b_1 b_j \right] , \qquad j = 2, \cdots, p+1 , \tag{49}
$$

$$
b_j^{(1)} = \nu_1 \left[ -b_1 a_{1j} + a_{11} b_j \right] , \tag{50}
$$

$$
c_1 = \frac{a_{11}}{\mu_1} \sqrt{\frac{k_1}{k'_1}} , \tag{51}
$$

with $(\mu_1, \nu_1)$ being the parameter pair which are still free to be chosen later. Note the close analogy of (46) - (51) to those of (23) - (28). Similarly, after the $i$-th rotation

$(1 < i \leq p)$, we have

$$k'_i = \frac{k_i a_{ii}^2 + k_q^{(i-1)} b_i^{(i-1)^2}}{\mu_i^2}, \tag{52}$$

$$k_q^{(i)} = \frac{k_i k_q^{(i-1)}}{\mu_i^2 \nu_i^2 k'_i}, \tag{53}$$

$$a'_{ii} = \mu_i, \tag{54}$$

$$a'_{ij} = \frac{1}{\mu_i k'_i} [\, k_i a_{ii} a_{ij} + k_q^{(i-1)} b_i^{(i-1)} b_j^{(i-1)} \,], \tag{55}$$

$$j = 2, \cdots, p+1,$$

$$b_j^{(i)} = \nu_i [\, -b_i^{(i-1)} a_{ij} + a_{ii} b_j^{(i-1)} \,], \tag{56}$$

$$c_i = \frac{a_{ii}}{\mu_i} \sqrt{\frac{k_i}{k'_i}}. \tag{57}$$

Finally after $p$ rotations are finished, (45) becomes

$$\begin{bmatrix} \sqrt{k'_1} & & & & \\ & \sqrt{k'_2} & & & \\ & & \ddots & & \\ & & & \sqrt{k'_p} & \\ & & & & \sqrt{k'_q} \end{bmatrix} \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1p} & a'_{1,p+1} \\ & a'_{22} & \cdots & a'_{2p} & a'_{2,p+1} \\ & & \ddots & \vdots & \vdots \\ & & & a'_{pp} & a'_{p,p+1} \\ 0 & 0 & \cdots & 0 & b_{p+1}^{(p)} \end{bmatrix} \tag{58}$$

which has the form of $\begin{bmatrix} R' & \mathbf{u}' \\ \mathbf{0}^T & v \end{bmatrix}$ in (42). The optimum residual $e$ in (43) now becomes

$$e = -\left( \prod_{i=1}^{p} \frac{a_{ii}}{\mu_i} \sqrt{\frac{k_i}{k'_i}} \right) \sqrt{k_q^{(p)}}\, b_{p+1}^{(p)}, \tag{59}$$

where $k_q^{(p)}$ is defined in (53). To further simplify the expression in (59), we notice

that $k_q^{(p)}$ can be computed recursively as follows,

$$
\begin{aligned}
k_q^{(p)} &= \left(\frac{\sqrt{k_p/k_p'}}{\mu_p \nu_p}\right)^2 k_q^{(p-1)} && (60)\\
&= \left(\frac{\sqrt{k_p/k_p'}}{\mu_p \nu_p}\right)^2 \left(\frac{\sqrt{k_{p-1}/k_{p-1}'}}{\mu_{p-1} \nu_{p-1}}\right)^2 k_q^{(p-2)} && (61)\\
&\vdots \\
&= \left[\prod_{i+1}^{p} \left(\frac{\sqrt{k_i/k_i'}}{\mu_i \nu_i}\right)^2\right] k_q \ , && (62)
\end{aligned}
$$

where (53) is used in the recursion.

With (62) substituted into (59), we have

$$
e \ = \ - \left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2 \, \nu_i} \frac{k_i}{k_i'}\right) \sqrt{k_q} \, b_{p+1}^{(p)} \ . \tag{63}
$$

Because $\sqrt{k_q}[b_1, b_2, \ldots, b_p, b_{p+1}] = [x_1, x_2, \ldots, x_p, y]$ is the appended new data row, we are certainly free to choose $k_q = 1$ to reduce the arithmetic complexity and simplify the expression in (63). Therefore, a Lemma on obtaining the sqrt-free optimum residual is given below.

**Lemma 1** *(Sqrt-free optimum residual)*

*The optimum residual $e$ can be computed with no square-root operations as follows:*

$$
e \ = \ - \left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2 \, \nu_i} \frac{k_i}{k_i'}\right) b_{p+1}^{(p)} \tag{64}
$$

*if $k_q$ is chosen to be unity.* □

McWhirter [11] successfully employed Gentleman's proposition [4] in computing the residual $e$ without sqrt operations. By choosing

$$
\mu_i \ = \ \nu_i \ = \ a_{ii} \ = \ 1, \ \ 1 \leq i \leq p \ , \tag{65}
$$

the optimum residual can be reduced to

$$e = -\left(\prod_{i=1}^{p} \frac{k_i}{k_i'}\right) b_{p+1}^{(p)} = -\left(\prod_{i=1}^{p} \bar{c}_i\right) b_{p+1}^{(p)} \qquad \text{(Gentleman/McWhirter)}, \tag{66}$$

where $\bar{c}_i$ is defined in (35). This result is again consistent with the work by McWhirter [11].

Another example can be taken from Hammarling's suggestion [8] as follows,

$$\mu_i = \frac{k_i a_{ii}^2 + k_q^{(i-1)} b_i^{(i-1)^2}}{k_i a_{ii}}, \tag{67}$$

$$i = 1, \cdots, p.$$

$$\nu_i = 1 / a_{ii}, \tag{68}$$

then it follows that $k_i' = k_i a_{ii}/\mu_i$ and the optimum residual is given by

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2 \nu_i} \frac{\mu_i}{a_{ii}}\right) b_{p+1}^{(p)} \tag{69}$$

$$= -\left(\prod_{i=1}^{p} \frac{1}{\mu_i \nu_i}\right) b_{p+1}^{(p)} \tag{70}$$

$$= -\left(\prod_{i=1}^{p} \frac{a_{ii}}{a_{ii}'}\right) b_{p+1}^{(p)} \qquad \text{(Hammarling)}. \tag{71}$$

# 5 Systolic Array Implementations

In implementating the above algorithms onto a VLSI structure, we need to take into account several factors which will affect its performance. Hardware complexity and/or speed consideration preempts square root opertions and even division operations are unfavorable. The required number of multiplications should also be minimized possibly. Another key issue is the parameters to be passed among the processing cells. This inter-cell transmitted data should be kept to its minimum.

An *optimum* mapping from any given algorithm to an architecture with concurrent processing cells is probably one of the solutions most sought after by many researchers. We will adopt the systolic triarray initiated by Gentleman and Kung [5] and McWhirter [11] as a framework and example for RLS filtering. Our approach presented here can be easily extended to other known or unknown sqrt-free algorithms belong to the $\mu\nu$ family. An optimum architecture can then be found from these mappings.

Figure 1 is a systolic triarray proposed by Gentleman and Kung and modified by McWhirter. By properly choosing the $\mu\nu$ parameters as given in Table 1, we are able to design the boundary and internal processing cells and also determine the required quantities for inter-cell transmission. If we take Gentleman's choice as an example and let $\mu_i = \nu_i = 1$ and $a_{ii} = a'_{ii} = 1, i = 1, 2, \ldots, p$, then (52) - (56) can be reduced to

$$k'_i = k_i + k_q^{(i-1)} b_i^{(i-1)^2}, \tag{72}$$

$$k_q^{(i)} = \frac{k_i}{k'_i} k_q^{(i-1)}, \tag{73}$$

$$a'_{ij} = \frac{k_i}{k'_i} a_{ij} + \frac{k_q^{(i-1)} b_i^{(i-1)}}{k'_i} b_j^{(i-1)}, \tag{74}$$

$$b_j^{(i)} = -b_i^{(i-1)} a_{ij} + b_j^{(i-1)}. \tag{75}$$

By defining $k_q^{(0)} = 1$, $\bar{c}_i = k_i/k'_i, \bar{s}_i = k_q^{(i-1)} b_1^{(i-1)}/k'_i$, and $\sigma_i = k_q^{(i)}$, we can easily see that the boundary cells are responsible for computing (72), (73), in addition to generating $\bar{c}_i$ and $\bar{s}_i$ for their right-hand-side regular cells. On the other hand, the regular cells, after obtaining the $\bar{c}_i$, $\bar{s}_i$, and $b_i^{(i-1)}$ can update its content and vertically

inputted data according to (74) and (75). The complete operations of these processing cells are given in Figure 1.

# 6 Conclusions

Planar (Givens) and hyperbolic rotations are the most commonly used methods in performing QRD up/downdating. Most of these rotation-based methods require explicit square-root computations, which are undesirable from the practical VLSI circuit design point of view. Since the square-root operation takes up much area and its computational time is slow (due to many iterations), the associated area/time efficiency is poor. Our work is the first effort to establish the basic understanding toward all known square-root-free QRD algorithms, from which the basic criterion is seen to be simple. We have shown that all the current known sqrt-free algorithms all belong to the $\mu\nu$ family. This unified approach also provides a fundamental framework for the square-root-free RLS algorithms which are essential for fast algorithms and practical VLSI implementations.

# References

[1] E. H. Bareiss, "Numerical solution of the weighted least squares problems by G-transformations," Tech. Report 82-03-NAM-03, Dept. of Elec. Engr. and Comp. Sci., Northwestern University, Evanston, IL, Apr. 1982.

[2] J. L. Barlow and I. C. F. Ipsen, "Scaled Givens rotations for the solution of linear least squares problems on systolic arrays," *SIAM J. Sci. Stat. Comput.,* Vol. 8, No. 5, pp. 716–733, Sept. 1987.

[3] M. J. Chen and K. Yao, "Comparison of QR least-squares algorithms for systolic array processing," *Proc. Conf. on Information Science and Systems,* pp. 683–688, Mar. 1988.

[4] W. M. Gentleman, "Least squares computations by Givens transformations without square roots," *J. Inst. Maths Applics,* Vol. 12, pp. 329–336, 1973.

[5] W. M. Gentleman and H. T. Kung, "Matrix triangularization by systolic arrays," *Proc. SPIE 298, Real-Time Signal Processing IV,* pp. 19–26, 1981.

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations,* Johns Hopkins Press, 2nd ed., 1989.

[7] J. Götze and U. Schwiegelshohn, "An orthogonal method for solving systems of linear equations without square roots and with few divisions," *Proc. IEEE ICASSP,* pp. 1298–1301, 1989.

[8] S. Hammarling, "A note on modifications to the Givens plane rotation," *J. Inst. Maths Applics,* Vol. 13, pp. 215–218, 1974.

[9] S. Kalson and K. Yao, "Systolic array processing for order and time recursive generalized least-squares estimation," *Proc. SPIE 564, Real-Time Signal Processing VIII,* pp. 28–38, 1985.

[10] F. Ling, D. Manolakis, and J. G. Proakis, "A recursive modified Gram-Schmidt algorithm for least-squares estimation," *IEEE Trans. on Acous., Speech, and Signal Processing,* Vol. ASSP-34, No. 4, pp. 829–836, Aug. 1986.

[11] J. G. McWhirter, "Recursive least-squares minimization using a systolic array," *Proc. SPIE 431, Real time signal processing VI,* pp. 105–112, 1983.

[12] I.K. Proudler, J.G. McWhirter, and T.J. Shepherd, "The QRD-basedleast squares lattice algorithm: some computer simulations using finite wordlength", Proc. IEEE ISCAS, pp.258-261, New Orleans, May 1990.

[13] F. Ling, "Efficient least squares lattice algorithm based on Givens rotations with systolic array implementation", Proc. IEEE ICASSP, pp.1290-1293, 1989.

**Figure Captions:**

**Fig.1** McWhirter's sqrt-free systolic triarray.

**Table 1** Comparisons of the SBHT and Givens rotation methods.

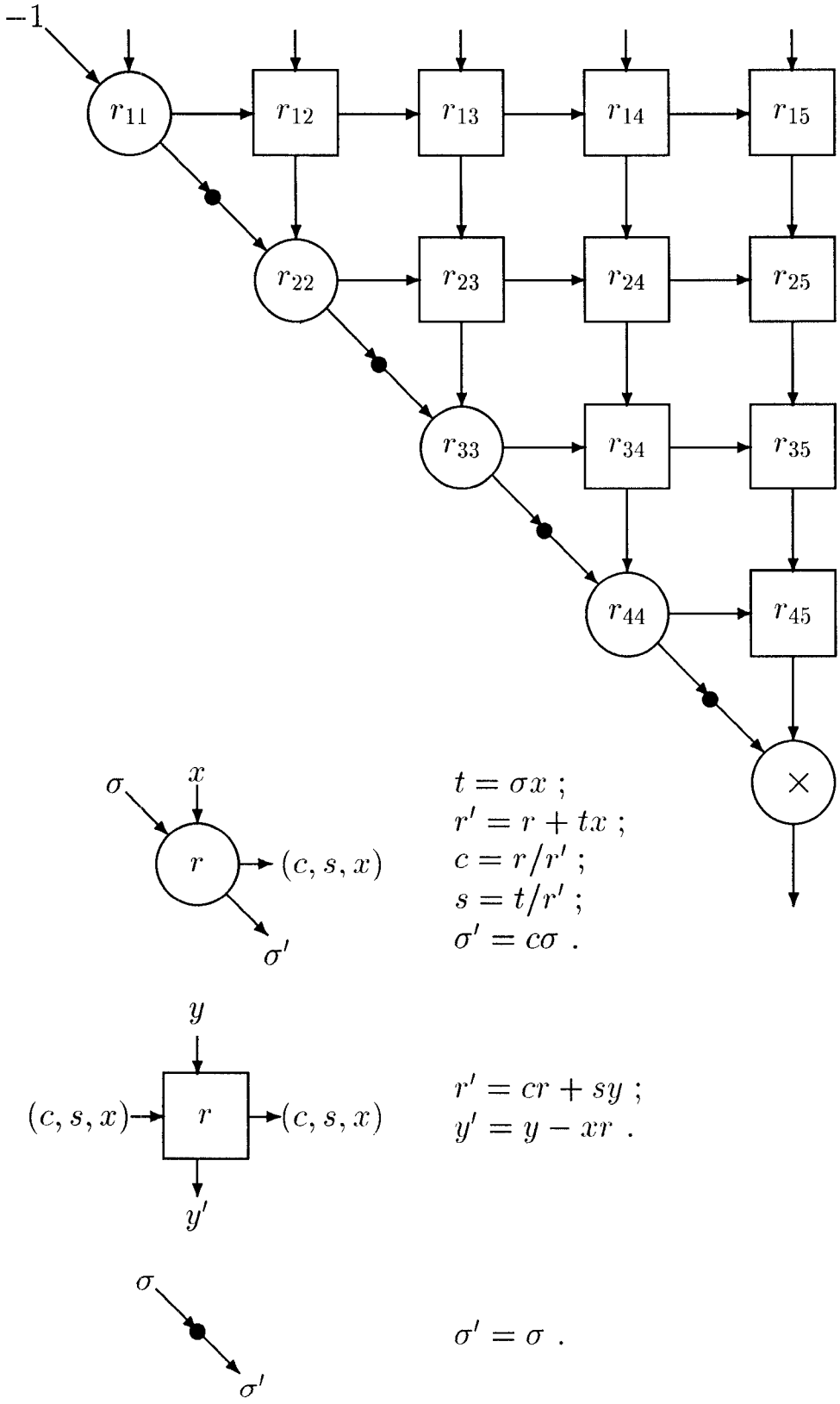


Figure 1: McWhirter's sqrt-free systolic triarray