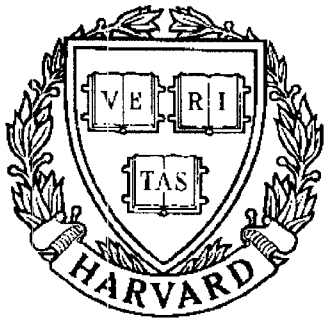
THESIS REPORT
Ph.D.

S Y S T E M S
R E S E A R C H
C E N T E R



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

Adaptive Array Systems Using QR-Based RLS and CRLS Techniques with Systolic Array Architectures

by C.-.F.T. Tang
Advisor: K.J.R. Liu

Ph.D. 91-5

ABSTRACT

Title of Dissertation: Adaptive Array Systems Using QR-Based RLS and CRLS Techniques with Systolic Array Architectures

Che-Fen Tom Tang, Doctor of Philosophy, 1991

Dissertation directed by: Dr. S. A. Tretter, Associate Professor
Dr. K.J. R. Liu, Assistant Professor
Systems Research Center
Electrical Engineering Department
University of Maryland
College park, Maryland 20742

In this dissertation the basic techniques for designing more sophisticated adaptive array systems are first developed. Then several systolic architectures based on numerically stable and computationally efficient algorithms are proposed for adaptive array systems. Compared to the existing architectures proposed elsewhere in the literature, our new systolic architectures are more efficient structures for real-time signal processing applications and VLSI hardware implementation. The reasons are: (1) the proposed systolic architectures are based on numerically stable and computationally efficient systolic algorithms, (2) there is no bottleneck in the whole architecture since QR decomposition by the square root free fast Givens method is used, (3) the whole architecture has a fully pipelined design since backward substitution is avoided, (4) it is a single fully pipelined open-loop system without any feedback arrangement, and (5) the systolic architectures function recursively to update the result for each new snapshot. Therefore, the new VLSI systolic architectures proposed in this dissertation using QR-recursive least squares (QR-RLS) and QR-constrained recursive least squares (QR-CRLS) techniques achieve minimal memory and maximal parallelism for real-time signal processing applications and VLSI hardware implementation.

Adaptive Array Systems Using QR-Based RLS and CRLS Techniques with Systolic Array Architectures

by
Che-Fen Tom Tang

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirement for the degree of
Doctor of Philosophy
1991

Advisory Committee:

Dr. K.J. R. Liu, Assistant Professor, Electrical Engineering, Advisor
Dr. S. A. Tretter, Associate Professor, Electrical Engineering, Advisor
Dr. A. Mahalanobis, Assistant Professor, Electrical Engineering
Dr. S. L. Koh, Professor, Associate Dean of College of Engineering
Dr. S. Peng, Assistant Professor, Computer Science
Dr. S. M. Yuen, Scientific Staff, GE Aerospace

To my wife: Betty Cheng

To my parents: F. Tang and P.S. Chang

for their love, support, inspiration

ACKNOWLEDGMENTS

I wish to thank my professors who have contributed to my dissertation. In particular, I thank my advisors Dr. S. A. Tretter for guiding me in writing and correcting this dissertation and Dr. K.J. R. Liu for his numerous discussions, suggestions, and inspiration which make this dissertation possible. I thank them for spending many hours helping me. It is my privilege to be their student and friend. I also wish to thank the other members of the dissertation committee, Dr. S. L. Koh, Dr. A. Mahalanobis, Dr. S. Peng, and Dr. S. M. Yuen for reading this dissertation and providing numerous helpful comments.

I would like to express my appreciation to Dr. Henry E. Lee of Westinghouse for introducing me to the area of systolic arrays. I would like to express my gratitude to Dr. Stanley M. Yuen of GE Aerospace for his constant help and valuable inputs. I gratefully acknowledge Dr. Y. J. Chen, Dr. C. I Chang, and Dr. C. Menyuk at UMBC for their financial support and help. Special thanks are due to Dr. Severino L. Koh, Associate Dean of College of Engineering, for his continuous financial support and encouragement.

This work was performed at Systems Research Center and Electrical Engineering Department, University of Maryland, College Park as well as at University of Maryland Graduate School, Baltimore. I appreciate the excellent research environment and computational facilities at University of Maryland.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

The problem of designing an adaptive array system is an important part of general radar and communication systems. The performance of conventional signal reception systems is sensitive to decreases in signal-to-noise (SNR) caused by undesired interference signals which may enter the system either by the beam pattern sidelobes or by the mainlobe. These interference signals which include waveforms from point sources in other than the look direction are lightning, jammers, noise from nearby vehicles, and localized incoherent clutter. Such SNR degradation may be further aggravated by antenna motion, poor siting conditions, multipath ray effects, and a constantly changing interference environment. As radar and communication traffic increases, the suppression of interference becomes more important in all applications. In radar, sonar, seismic, and communication systems, adaptive arrays can be utilized to

preserve the desired signal in the presence of interference signals. Moreover, adaptive array systems have the ability to automatically sense the presence of interference signals and to suppress precisely these interference signals while simultaneously enhancing the desired signal without prior knowledge of the signal/interference environment. Therefore, an adaptive array system consists of an array of sensors and a real-time computational processor to automatically adjust the array sensitivity beam pattern so that the array performance is improved. As a result, adaptive array systems offer enhanced reliability compared to that of conventional multi-sensor systems [1].

In designing adaptive array systems, the recursive least-squares (RLS) and constrained recursive least-squares (CRLS) approaches are adopted to achieve faster convergence compared to that of least-mean-square (LMS) approaches because the RLS and CRLS algorithms utilize all the information contained in the input data from the start of the adaptation up to the present whereas the LMS algorithm does not. The price to be paid for this improvement is increased complexity. Algorithms based on the RLS or CRLS criterion are considered to be the best candidates for adaptive filters. They have two fundamental properties [2]. First, the LS estimate is the best linear unbiased estimate under the condition of white with zero mean for the measurement error process. That is, it has minimum variance among the class of all linear unbiased estimates. Second, if the noise involved satisfies the independent and

Gaussian condition, the RLS or CRLS estimate is the most efficient estimate among all unbiased estimates. In other words, it achieves the Cramer-Rao bound. In the radar and communication community, the conventional approach for solving RLS or CRLS problems generally uses direct sample matrix inversion (SMI). Although the direct SMI solution is straightforward, it has two major disadvantages. One of the disadvantages is that the SMI method has undesirable numerical characteristics when the sample covariance matrix is ill conditioned. Extremely high arithmetic precision is required when using the direct SMI method to prevent numerical instability. Another disadvantage is that the SMI algorithm can not be implemented as parallel/pipelined array processors for real-time signal processing applications. To remedy the problems of ill-conditioning and VLSI implementation, a family of algorithms based on orthogonalization techniques can be used. They are the Givens, modified Givens, Householder, Gram-Schmidt, and modified Gram-Schmidt methods. The QR-based algorithms which deal directly with the observed data matrix achieve the requirements of computational efficiency, robust numerical stability and VLSI parallel/pipelined architecture implementation. Therefore, the subject of mapping orthogonalization algorithms onto systolic arrays has become very important for real-time signal processing applications with VLSI hardware.

The importance of developing numerically stable and computationally effi-

cient systolic algorithms as well as mapping them into VLSI parallel/pipelined architectures will be demonstrated in this dissertation. Modern signal processing algorithms, especially adaptive arrays and multichannel algorithms, are well structured to share the common attributes of regularity, recursiveness, and local communication. These properties are effectively exploited in VLSI parallel/pipelined architectures. In VLSI hardware devices, memory and processing power are relatively cheap and the main emphasis of the design is concentrated on reducing the overall interconnection complexity and on keeping the overall architecture highly regular, parallel, and pipelined. Moreover, these devices offer massive concurrent computing which is essential to real-time high throughput signal processing [7,30,44,45,46,48].

Among existing algorithms and their related systolic architectures, the QR-based RLS and CRLS methods have proven to be very useful and effective in adaptive array systems used for system identification, channel equalization, adaptive antenna arrays, spectrum estimation, etc. [1,2,3,4,6,7,8,9,10,11,12,14, 15,16,17,22,23,24,27,28,31,32,33,34,35,36,37,45,49,63,61,62,63,64,65,66,68,69,72, 73]. In this dissertation, the design of numerically stable and computationally efficient systolic algorithms using QR-based RLS and CRLS techniques with implementation by VLSI systolic array architectures is the major subject. The dissertation begins with a design for a parallel/pipelined weight extraction systems based on systolic RLS and CRLS techniques. Then an er-

ror in McWhirter's MVDR systolic array processors is pointed out and a new design for MVDR systolic array processors is presented. Finally systolic array architecture design using the Householder transformation which is the best computational efficient and numerical stable method among the family of QR decomposition for adaptive array system is discussed.

This dissertation is organized as follows. In Chapter 2, the theory for the QR decomposition-based Givens and fast Givens methods is stated in detail and the new developments of the systolic array matrix computations are described for designing matrix-based parallel signal processing. Chapter 3 is divided into two parts. The first part consists of the QR-based RLS weight extraction algorithm and its related systolic array architecture. The second part consists of the QR-based CRLS weight extraction algorithm and its systolic array architecture. Since the proposed systolic architectures based on numerically stable algorithms are the only known fully pipelined structures without the need for a back substitution processor, they are considered to be the best design.

In Chapter 4 we point out an error in McWhirter's MVDR systolic algorithm and architecture and propose the correct MVDR systolic algorithm and architecture. In Chapter 5, the most computationally efficient and numerically stable Householder method in the family of QR decompositions with the systolic array architecture is presented and the application to an RLS array

system is described along with systolic array implementation. Finally, Chapter 6 is devoted to conclusions and suggestions for future research.

Chapter 2

Systolic Array Linear Algebra Processing

2.1 Introduction

In the emerging field of *algorithmic engineering* introduced by McWhirter [11], the hybrid disciplines of designing numerically stable parallel algorithms suitable for parallel computation and mapping them onto VLSI systolic architectures to achieve high throughput rates and VLSI hardware implementation are demanded for sophisticated, high performance real-time modern signal processing. In real-time modern signal processing applications, numerically reliable and computationally efficient algorithms for techniques such as recursive least squares estimation (RLS), constrained recursive least squares estimation

(CRLS), solving linear systems, and performing singular value decomposition are required. Furthermore, in these applications it is also necessary to design a highly parallel/pipelined structure for the use in parallel supercomputers and for implementation by systolic processors.

In this chapter some key processors are developed as the basic tools for designing sophisticated adaptive array systems. Moreover, the parallel/pipelined techniques invented here make it possible to design more advance adaptive array systems such as the RLS and CRLS arrays studied in this dissertation. A detailed description of linear-algebra-based parallel algorithms with systolic array processors is provided in the following sections. In the first two sections, QR decompositions based on the Givens and fast Givens method with systolic array processors are described in detail. In the third section a brief description of how a QR decomposition updates a system is presented. In the rest of the sections the important matrix computations used in the array systems are studied and the associated systolic array processors are designed. By using the parallel/pipelined techniques developed in this chapter, we are able to design more complicated adaptive array systems.

2.2 Systolic Array QR Decomposition by Complex Givens Method

An $N \times N$ data matrix $X(n)$ can be reduced to an upper triangular matrix $R(n)$ by a transformation of the form

$$\begin{bmatrix} R(n) \\ 0 \end{bmatrix} = Q(n)X(n) \tag{2.1}$$

where $R(n)$ is the upper triangular matrix

$$\begin{bmatrix} r_{11}(t_n) & r_{12}(t_n) & r_{13}(t_n) & \cdots & r_{1N}(t_n) \\ 0 & r_{22}(t_n) & r_{23}(t_n) & \cdots & r_{2N}(t_n) \\ 0 & 0 & r_{33}(t_n) & \cdots & r_{3N}(t_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{NN}(t_n) \end{bmatrix},$$

and $Q(n)$ is an $n \times N$ unitary matrix.

2.2.1 Givens Algorithm

To illustrate how a complex Givens orthogonalization triangularizes the data matrix X, we apply a 2 by 2 unitary matrix to the two rows of X to zero out an element.

$$\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1\,N} \\ x_{21} & x_{22} & \cdots & x_{2\,N} \end{bmatrix} = \begin{bmatrix} r'_{11} & r'_{12} & \cdots & r'_{1\,N} \\ 0 & x'_{22} & \cdots & x'_{2\,N} \end{bmatrix} \tag{2.2}$$

where

$$x'_{11} = \sqrt{r_{11}^2 + |x_{21}|^2} \tag{2.3}$$

$$c = \frac{r_{11}}{x'_{11}} \tag{2.4}$$

$$s = \frac{x_{21}}{x'_{11}} \tag{2.5}$$

and

$$r'_{1j} = cr_{1j} + s^* x_{2\,j} \qquad for\ j = 2, \cdots, N. \tag{2.6}$$

$$x'_{2j} = -sr_{1j} + cx_{2j} \qquad for\ j = 2, \cdots, N. \tag{2.7}$$

Note that c, r'_{11}, *and* r_{11} are real and s, x'_{1j}, x'_{2j} are complex for $j = 2, \cdots, N$. In addition, the diagonal elements of R can be made real because X can be also expressed as $(QD)(D^{-1}R)$ which is also a QR decomposition when D is any unitary diagonal matrix [29].

2.2.2 Givens Based Systolic Array Processors

A Givens upper triangular systolic array processor is illustrated in Figure 2.1. The symbols for the boundary and internal cells for a Givens upper triangular structure are depicted in Figure 2.2. In Table 2.1 the algorithms for the boundary cell and internal cell are described.

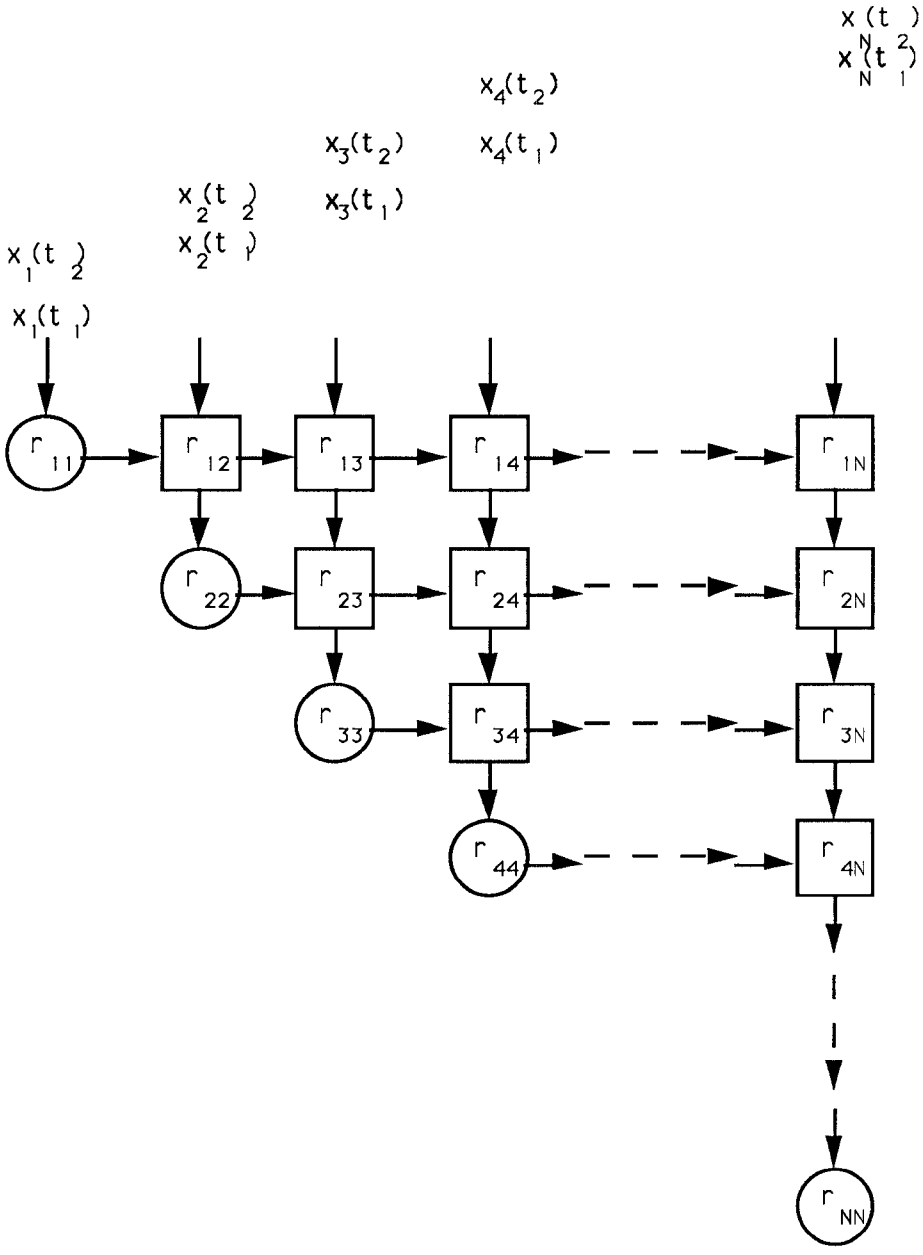


Figure 2.1: Givens Based Upper Triangular Systolic Array

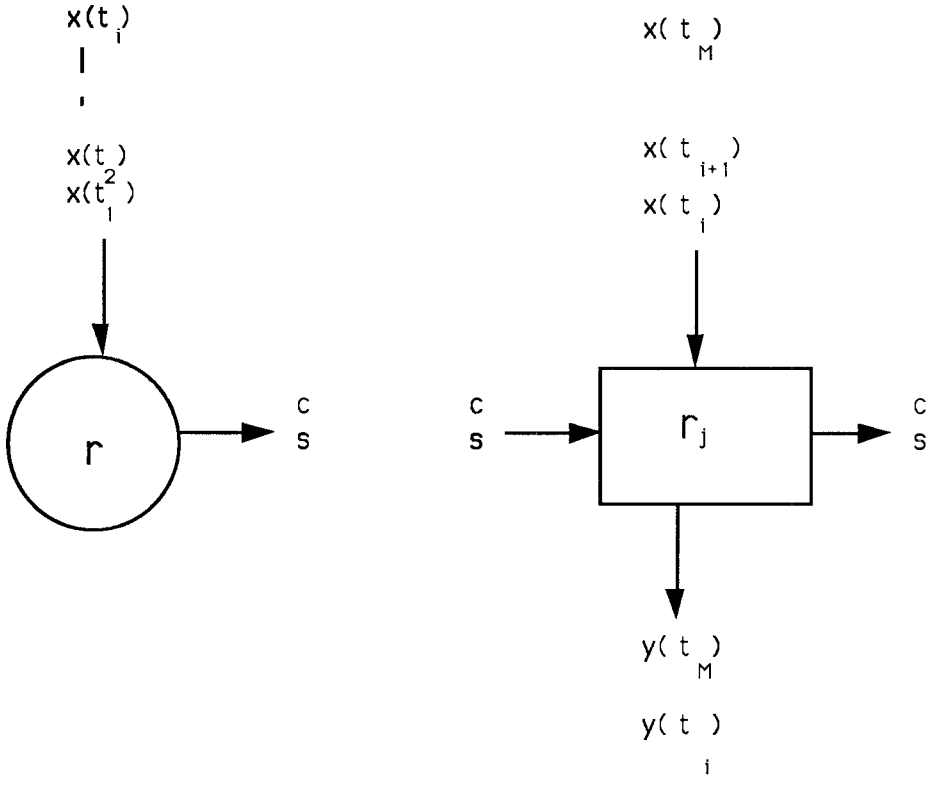


Figure 2.2: The Boundary and Internal Cells of Givens Based Systolic Array

Boundary Cell :	*Internal Cell* :
$d \leftarrow \sqrt{r^2 + \lvert x(t_i)\rvert^2}$	$y(t_i) \leftarrow -sr + cx(t_i)$
$c \leftarrow \frac{r}{d}$	$r \leftarrow cr + s^* x(t_i)$
$s \leftarrow \frac{x(t_i)}{d}$	
$r \leftarrow d$	

Table 2.1: Givens Algorithms for the Boundary Cell and Internal Cell

2.3 Systolic Array QR Decomposition by Complex Fast Givens Method

One of the important subjects in implementing QR decomposition onto VLSI hardware is to avoid the square root. Since the computation of the square root is complicated, it creates the bottleneck in the QR systolic processor. To speed up the system, a square root free version of Givens method is necessitated and leads to the name *fast Givens method* as first proposed by Gentleman [19] and Hammarling [20]. Since the square root can be stored in the diagonal elements, the upper triangular matrix $D^{\frac{1}{2}}\overline{R}_{upper}$ is expressed as follows.

$$D^{\frac{1}{2}}(n)\overline{R}(n) = Q(n)X(n) \tag{2.8}$$

where $D(n)$ is a N by N diagonal matrix which has form

$$\begin{bmatrix} d_1(t_n) & 0 & \cdots & 0 \\ 0 & d_2(t_n) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_N(t_n) \end{bmatrix},$$

and $\overline{R}(n)$ is an upper triangular matrix which is given

$$\begin{bmatrix} 1 & r_{12}(t_n) & r_{13}(t_n) & \cdots & r_{1N}(t_n) \\ 0 & 1 & r_{23}(t_n) & \cdots & r_{2N}(t_n) \\ 0 & 0 & 1 & \cdots & r_{3N}(t_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

.

2.3.1 Fast Givens Algorithm

The following example of 2×2 matrix will show how the fast Givens method works. A complex fast Givens method triangularizes the data matrix as follows: First let

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \end{bmatrix} = \begin{bmatrix} \sqrt{d} & 0 \\ 0 & \sqrt{\delta} \end{bmatrix} \begin{bmatrix} 1 & r_2 & \cdots & r_N \\ y_1 & y_2 & \cdots & y_N \end{bmatrix} \tag{2.9}$$

where $d = r_{11}^2$ and $\delta = k^2$

$r_j = \frac{x_{1j}}{x_{11}}$ for $; j = 2, \cdots, N.$

$y_j = \frac{x_{2j}}{k}$ for ; $j = 2, \cdots, N.$

Then, the complex fast Givens method applied to the data matrix to zero an element is [19,20]

$$\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \begin{bmatrix} \sqrt{d} & 0 \\ 0 & \sqrt{\delta} \end{bmatrix} \begin{bmatrix} 1 & r_2 & \cdots & r_N \\ y_1 & y_2 & \cdots & y_N \end{bmatrix} = \begin{bmatrix} \sqrt{d'} & 0 \\ 0 & \sqrt{\delta'} \end{bmatrix} \begin{bmatrix} 1 & r_2' & \cdots & r_N' \\ 0 & y_2' & \cdots & y_N' \end{bmatrix} \tag{2.10}$$

An analogous algorithm for fast Givens orthogonalization is given as

$$d' = d + \delta |y_1|^2 \tag{2.11}$$

$$\delta' = \frac{d\delta}{d'} \tag{2.12}$$

$$c' = y_1 \tag{2.13}$$

$$s' = \frac{\delta}{d'} y_1 \tag{2.14}$$

and

$$r_j' = r_j + s'^* y_j' \qquad for\ j = 2, \cdots, N. \tag{2.15}$$

$$y_j' = -c' r_j + y_j \qquad for\ j = 2, \cdots, N. \tag{2.16}$$

2.3.2 Fast Givens Based Systolic Array Processors

The upper triangular systolic array processor based on the fast Givens technique is illustrated in Figure 2.3. The symbols for the boundary and internal

Boundary Cell	*Internal Cell* :
$k \leftarrow r + \delta_{in}\lvert x^2(t_i)\rvert^2$	$y(t_i) \leftarrow -cr + x(t_i)$
$\delta_{out} \leftarrow \frac{r\delta}{k}$	$r \leftarrow r + s^* y(t_i)$
$c \leftarrow x(t_i)$	
$s \leftarrow \frac{\delta x(t_i)}{k}$	
$r \leftarrow k$	

Table 2.2: Fast Givens Algorithms for the Boundary Cell and Internal Cell

cells for this upper triangular structure are described in Figure 2.4. The algorithms for the boundary cell and internal cell are stated in Table 2.2.

2.4 Systolic Array for Forward Substitution

In [21] the computation of $R^{-H}X$ by the Comon-Robert's algorithm is carried out in two steps when an upper triangular matrix R and a matrix X are given. In the first step, a matrix R^{-1} is computed when the matrix R is fed into the systolic array. In the second step, the vector X^H is given as input to compute $X^H R^{-1}$. As a result, the complex conjugate of $R^{-H}X$ is computed by a systolic parallelogram structure. In [14,22] the same computation of $R^{-H}X$ requires only one step with the matrix R prestored in the systolic array and the matrix X as input fed into the array. A more detailed description of the computation

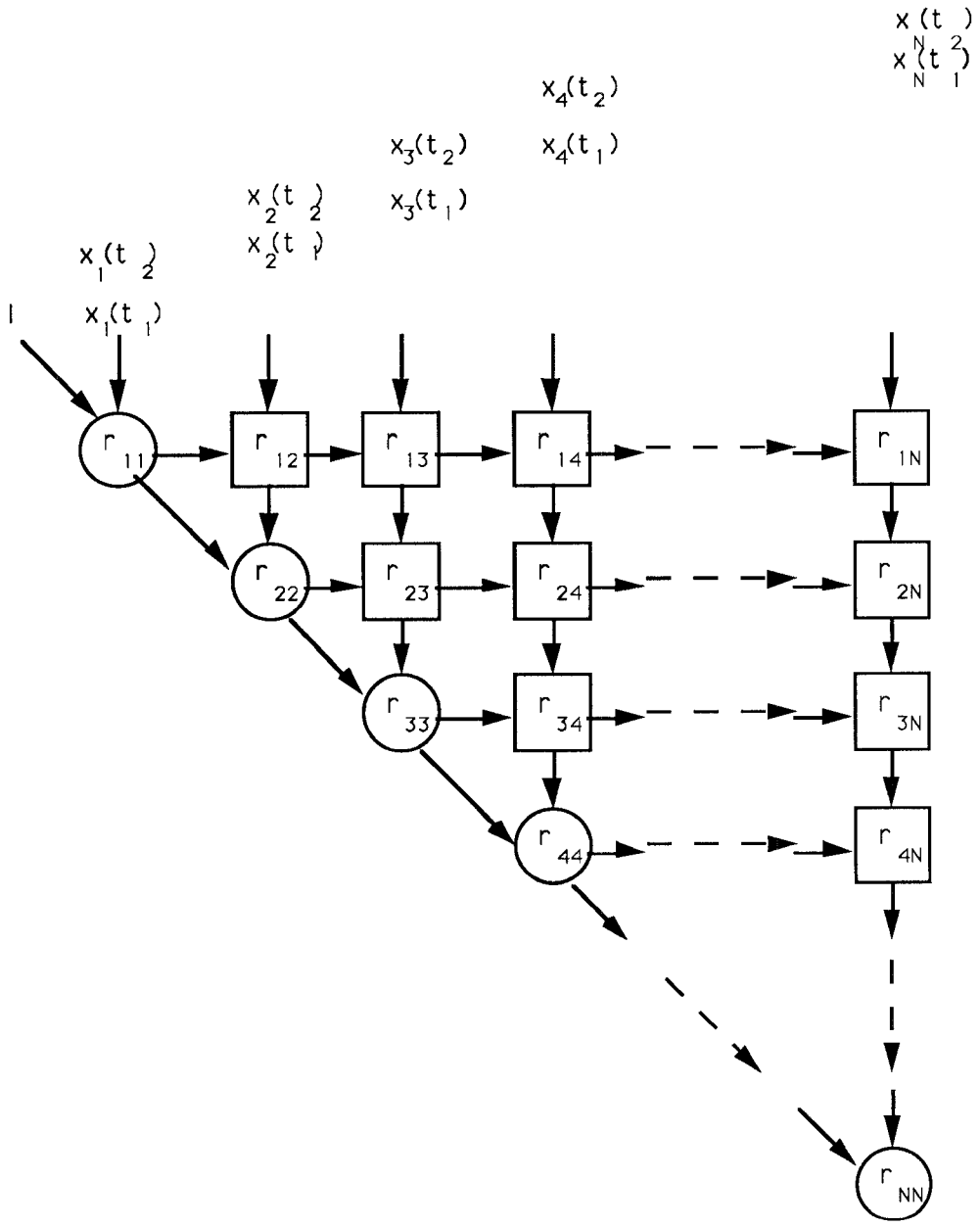


Figure 2.3: Fast Givens Based Upper Triangular Systolic Array

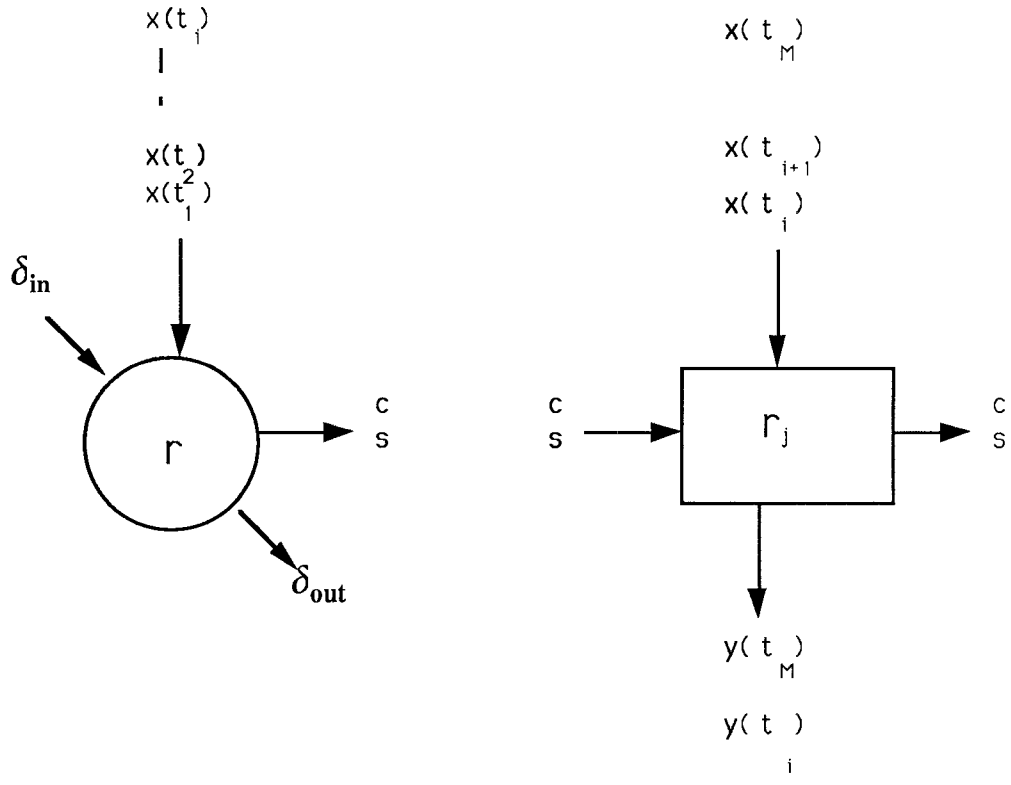


Figure 2.4: The Boundary Cell and Internal Cell for Fast Givens Based Systolic Array

of $R^{-H}X$ can be easily generalized from Liu-Yao's algorithm [22]. We give below a brief description of the algorithm for computation $R^{-H}X$.

Assume $r_{ij} = (R)_{ij}$ and $r'_{ij} = (R^{-H})_{ij}$, where R is an upper triangular matrix and R^{-H} is a lower triangular matrix. It is known that the relationship between r_{ij} and r'_{ij} is

$$r'_{ij} = \begin{cases} \frac{1}{r^*_{ii}} & \text{for } i = j \\ -\sum_{k=i}^{j-1} r'_{ik} \frac{r^*_{kj}}{r^*_{jj}} & \text{for } n \geq i > j \end{cases} \tag{2.17}$$

The following equations show how to compute $Y = R^{-H}X$ recursively where R^{-H} is a $n \times n$ matrix and X is a $n \times m$ matrix. Assume $y_{ij} = (R^{-H}X)_{ij}$ where $x_{ij} = (X)_{ij}$. Hence, y_{ij} is given by

$$y_{ij} = \sum_{l=1}^{i} r'_{il} x_{lj} \quad for \quad i = 1, \cdots, n. \quad and \quad j = 1, \cdots, m. \tag{2.18}$$

Since we want to use R and X to compute $R^{-H}X$, let us express y_{ij} in terms of r_{ij} and x_{ij}. By substituting Equation 2.17 into Equation 2.18 and rewriting the equation, we have

$$y_{ij} = \frac{1}{r^*_{jj}}(x_{ij} - \sum_{k=1}^{j-1} y_{ik} r^*_{kj}) \tag{2.19}$$

The pipelined data-parallel algorithm presented in Table 2.3 computes $Y = R^{-H}X$ where Y is $n \times m$ matrix, R is $n \times n$ upper triangular matrix, and X is $n \times m$ matrix. A systolic parallelogram structure for computing $R^{-H}X$ is illustrated in Figure 2.5.

An Algorithm for Computing $Y = R^{-H} X$

for $i = 1$ *to* n

begin

$y_{i1} = \frac{1}{r^*_{11}} x_{i1}$

in parallel for $j = 2$ *to* m

begin

$z_{ij} = x_{ij}$

in parallel for $k = 1$ *to* $j - 1$

$z_{ij} = z_{ij} - y_{ik} r^*_{kj}$

$y_{ij} = \frac{z_{ij}}{r^*_{jj}}$

end

end

Table 2.3: The Parallel/Pipelined Algorithm for Forward Substitution

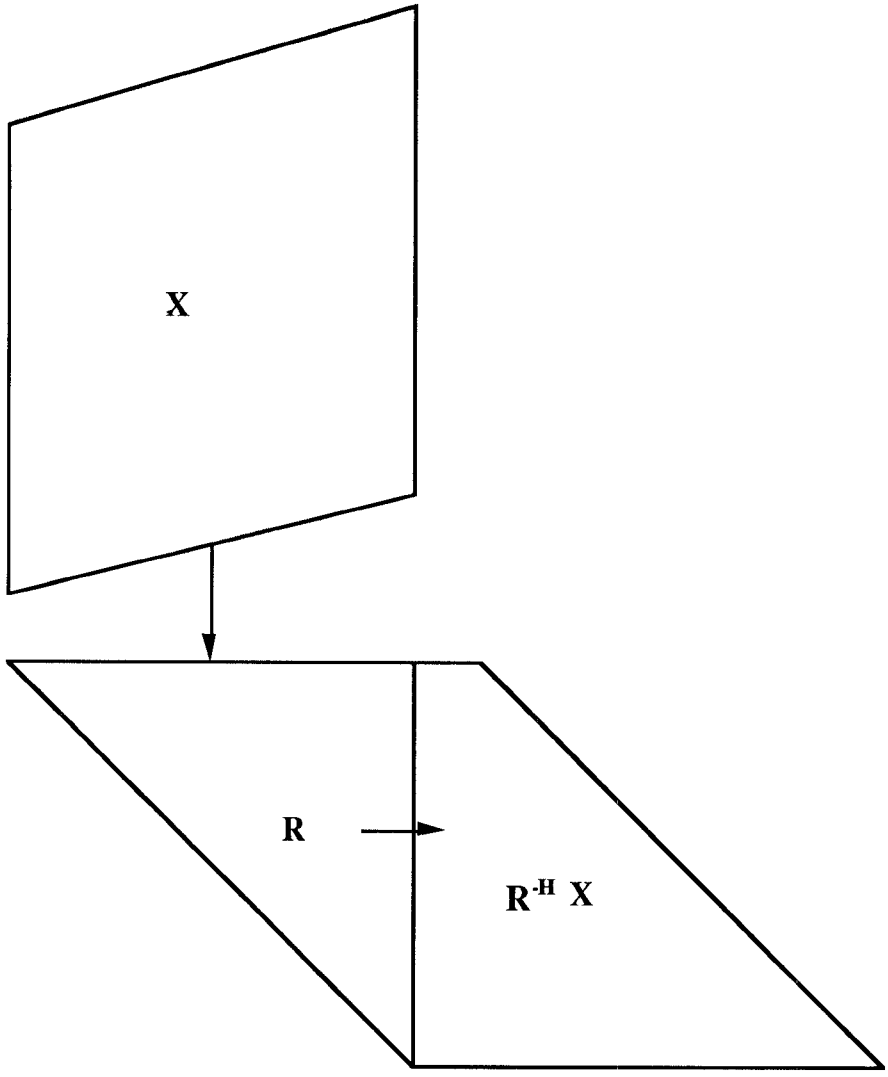


Figure 2.5: Systolic Array Processors for Forward Substitution

2.5 Systolic Array for Backward Substitution

In adaptive array systems, we have to compute backward substitution for the system. Since the vector $\underline{z}$ and the lower triangular matrix R^{-H} are the given data, the question now is how to design a systolic structure to compute $R^{-1}\underline{z}$, backward substitution, by using the given vector $\underline{z}$ and the given lower triangular matrix R^{-H}. Fortunately, this can be easily carried out on a systolic array recursively. A more detailed description of this computation is given and the systolic structure is described below.

Assume that a 3×3 matrix R^{-1} has the form

$$R^{-1} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ 0 & x_{22} & x_{23} \\ 0 & 0 & x_{33} \end{bmatrix} \tag{2.20}$$

and a 3×1 vector $\underline{z}$ is given by

$$\underline{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{2.21}$$

The computation of $R^{-1}\underline{z}$ is

$$R^{-1}\underline{z} = \begin{bmatrix} x_{11}z_1 + x_{12}z_2 + x_{13}z_3 \\ x_{22}z_2 + x_{23}z_3 \\ x_{33}z_3 \end{bmatrix} \tag{2.22}$$

Let us employ a lower triangular matrix R^{-H} to carry out the computation of $R^{-1}\underline{z}$. Then, R^{-H} has the form

$$R^{-H} = \begin{bmatrix} x_{11}^* & 0 & 0 \\ x_{12}^* & x_{22}^* & 0 \\ x_{13}^* & x_{23}^* & x_{33}^* \end{bmatrix} \tag{2.23}$$

In Table 2.4, a pipelined data-parallel algorithm for computing $R^{-1}\underline{z}$ is described. The systolic array shown in Figure 2.6 is designed by concurrently sending each element of the vector $\underline{z}$ to multiply the complex conjugate of each element of the matrix R^{-H} and then by summing them together to obtain the vector $R^{-1}\underline{z}$.

An Algorithm for Computing $R^{-1}\underline{z}$

in parallel for $i = 1$ *to* m, $j = 1$ *to* m

begin

$temp(i,j) = z(j) * x^*(i,j)$

end in parallel

in parallel for $i = 1$ *to* m

begin

$w(i) = 0$

in parallel for $j = 1$ *to* m

begin

$w(i) = w(i) + temp(i,j)$

end in parallel

end in parallel

Table 2.4: The Parallel/Pipelined Algorithm for Backward Substitution

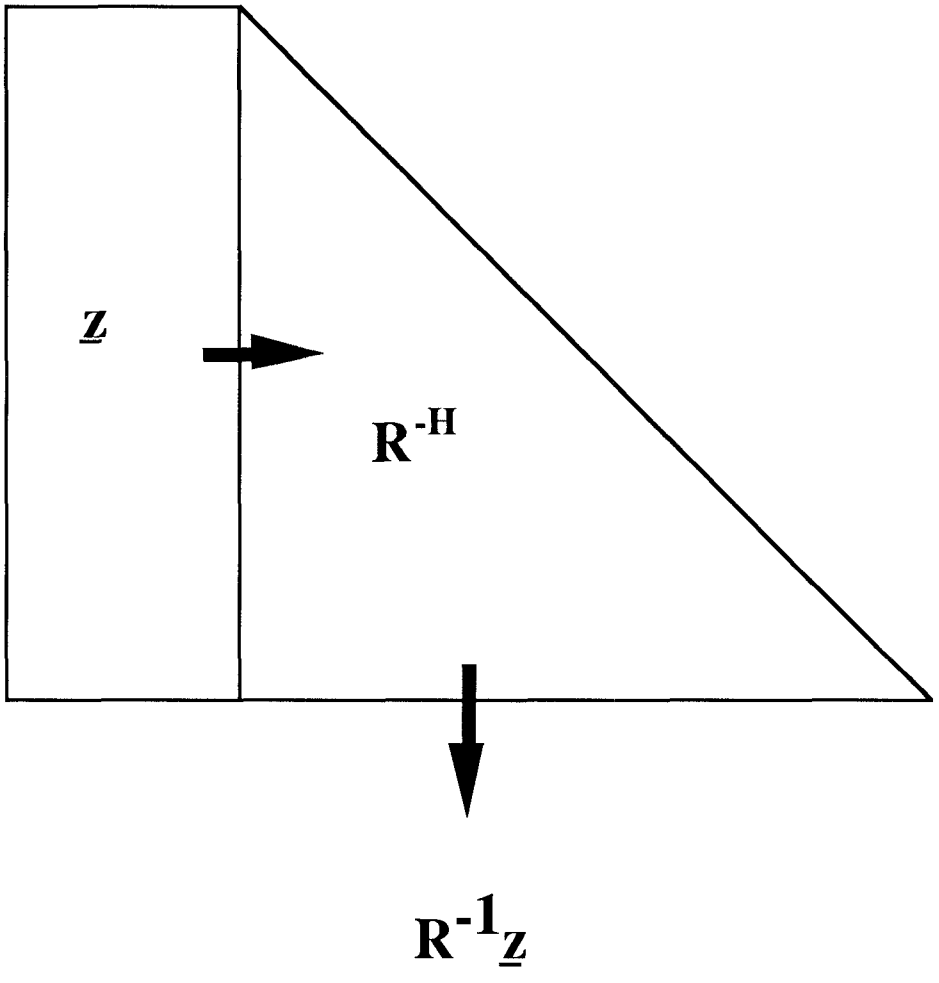


Figure 2.6: Systolic Array Processors for Backward Substitution

Chapter 3

Parallel/Pipelined Weight Extraction for RLS and CRLS Adaptive Array Systems

3.1 Introduction

The problem of parallel/pipelined weight extraction for systolic adaptive beamforming systems has been the subject of intense research since the first well known work of Gentleman and Kung on recursive least squares (RLS) systolic arrays [6,7]. As shown in Figure 3.1, although QR-updates are pipelined on a triangular array proposed in [6], a fully parallel/pipelined weight extraction from the RLS systolic array consists of the two separate steps of QR-updates

and backsolve and has been shown to be unrealizable.

A major issue in implementing the algorithm for adaptive beamforming by systolic array processors is to design a single fully pipelined structure. The critical obstruction appears because the process of QR-updates runs from the upper-left corner to lower-right corner and the process of the backsolve runs in exactly the opposite direction as pointed out in [8]. Much research has been done on this subject recently [9,10,11]. In [9], Hudson and Shepherd proposed a theoretically equivalent open-loop system and a closed-loop system for parallel weight extraction. Compared to the unstable theoretically equivalent open-loop system, a Kalman closed-loop feedback structure shown in Figure 3.2 which consists of a systolic QR decomposition post-processor to compute least squares weighting vector is considered to be more promising in systolic implementation. However, it is shown in Figure 3.2 that the parallel RLS weight extraction system proposed is not efficient for VLSI hardware implementation. The major hurdles are (1) this system, which requires two modes to update the data and to freeze the updated data for computing the weight vector error at same time, has a significant problem in obtaining the instant weight vector recursively, and (2) the feedback configuration may create serious routing problems in VLSI hardware implementation.

A brief description of Hudson-Shepherd's algorithm is presented next using the same notation as in Section 3.2. The associated weight vector update called

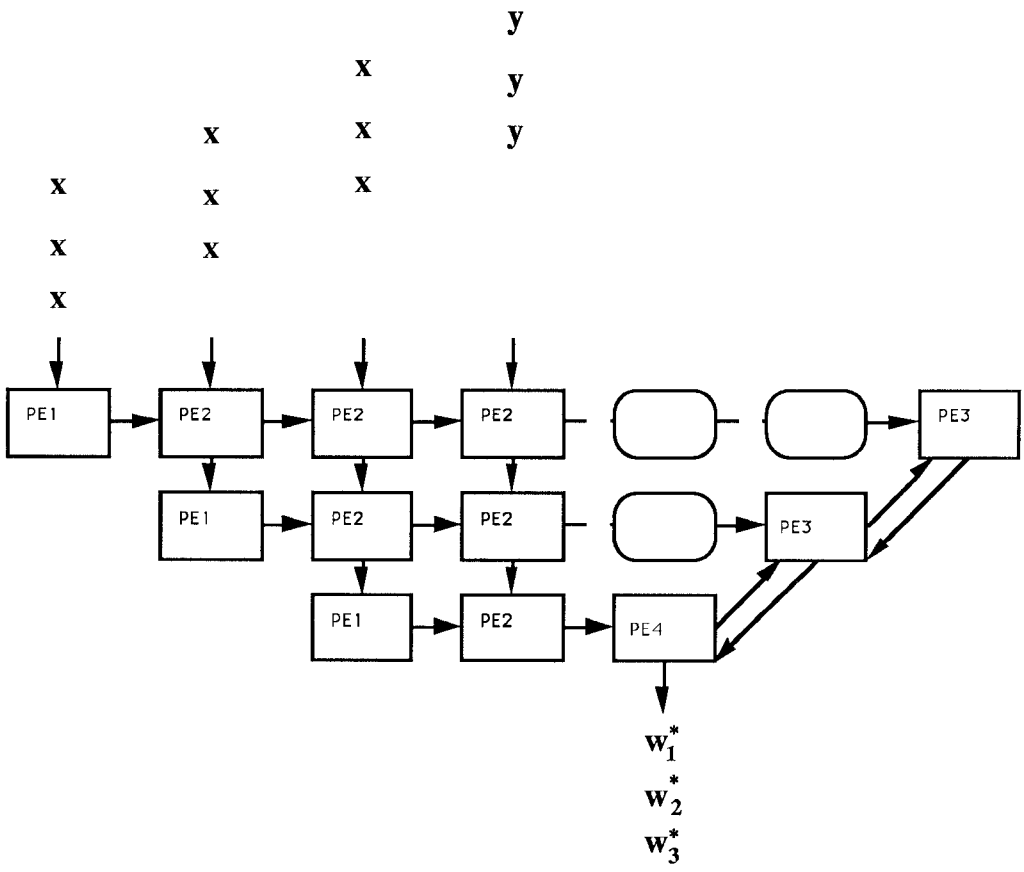


Figure 3.1: Gentleman and Kung's Systolic Architecture [6]

the "Kalman state update" has a form which is adopted from a recent paper [9]. The equation for using the Kalman state update method to update the RLS weighting vector is

$$\underline{w}^H(n) = \underline{w}^H(n-1) - M^{-1}(n)\underline{x}^T(t_n)e_{n/(n-1)} \tag{3.1}$$

where M, the data covariance matrix, is

$$M = X^H X$$

$e_{n/(n-1)}$, the residual error, is given by

$$e_{n/(n-1)} = \underline{x}^T(t_n)\underline{w}(n-1) - \underline{z}(n),$$

and $\underline{x}^T(t_n)$ is given as a input data vector. By applying QR decomposition to the observed data X, Equation 3.1 can be rewritten as follows:

$$\underline{w}^H(n) = \underline{w}^H(n-1) - R^{-1}(n)R^{-H}(n)\underline{x}^T(t_n)e_{n/(n-1)} \tag{3.2}$$

According to Figure 3.2, a vector $\underline{f}(n)$ and a vector $\underline{g}(n)$ are defined as follows:

$$\underline{f}(n) = R^{-H}(n)\underline{x}^T(t_n) \tag{3.3}$$

and

$$\underline{g}(n) = R^{-1}(n)\underline{f}(n) \tag{3.4}$$

It can be seen from Equation 3.3 that $\underline{x}^T(t_n)$ used to update the upper triangular matrix $R(n)$ in one mode is employed again as the input sent to the

updated matrix $R(n)$ for computing the vector $\underline{f}(n)$ in a different mode. As a consequence, Hudson-Shepherd's algorithm is not suitable for real-time application and VLSI hardware implementation.

In [11] McWhirter introduced a parallelogram fixed matrix operator structure for the parallel weight extraction problem shown in Figure 3.3. As mentioned by McWhirter, however, to update his RLS systolic array and to avoid freezing the array, a more complicated post-processor cell is required. As described in Figure 3.3, McWhirter's RLS weight extraction system does not function recursively to update the weight vector since the system requires freezing the array to compute the weight vector. By improving McWhirter's parallel weight extraction fixed operator algorithm and adding the updating processor to avoid freezing the array, a new algorithm will be derived to update the whole parallelogram systolic arrays and at same time to compute the RLS adaptive weight vector recursively.

In a recent paper [12], a numerically stable and computationally efficient algorithm for weight extraction for a constrained recursive least squares (CRLS) problem has been described by Schreiber. Although the algorithm shown in [12] has robust numerical properties, it is difficult to arrange the whole algorithm into a single fully pipelined structure as pointed out in [28,14]. The difficulty, which is the same as that in the RLS case, arises because the CRLS algorithm consists of several steps particularly involving the backsolve step.

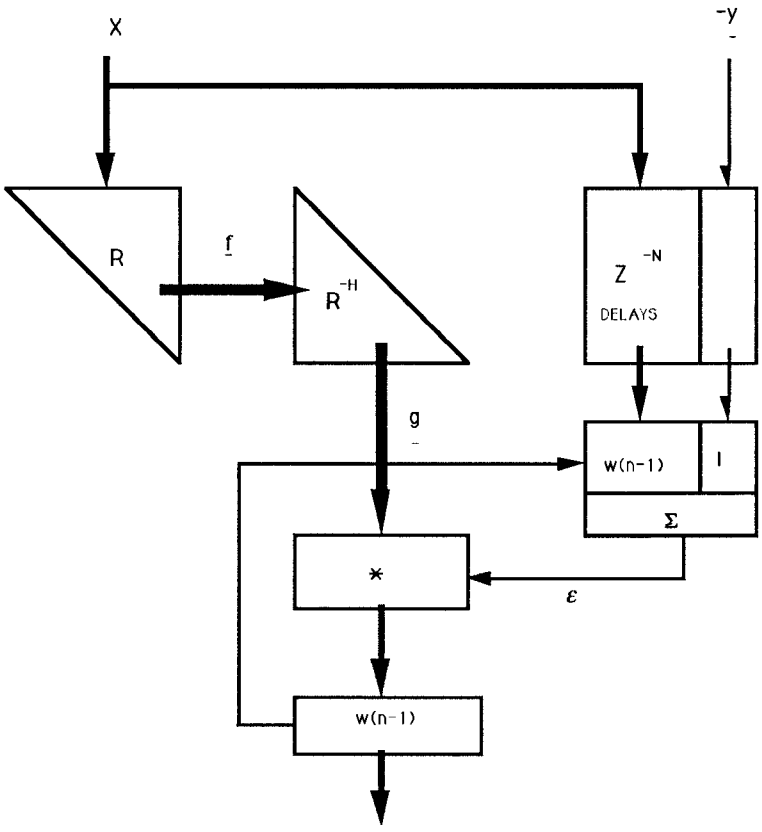


Figure 3.2: Hudson-Shepherd's Parallel Weight Extraction Structure [9]

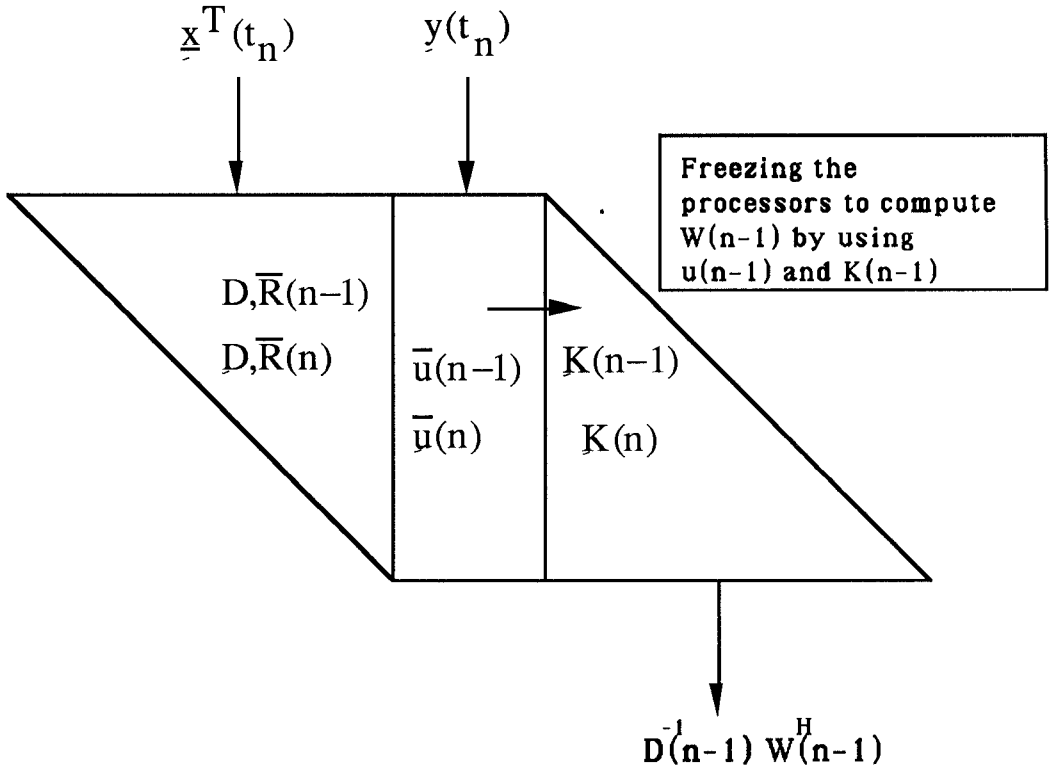


Figure 3.3: McWhirter's Parallel Weight Extraction Structure by Using Fixed Matrix Operator [11]

Many CRLS systolic array structures proposed in [28,14,15] are designed to avoid the extra backsolve processor for computing the residual. Unfortunately, for the problem of parallel/pipelined weight extraction, very little has been done in implementing the CRLS algorithm into a single fully pipelined systolic array structure without the need of the backsolve processor. In recent research [16,17] Owsley developed an adaptive CRLS beamformer with a systolic array implementation using Schreiber's algorithm for parallel weight extraction. Nevertheless, Owsley's CRLS systolic array structure which consists of several block processors including a backsolve processor has been shown to be unpipelinable.

In this chapter a single fully parallel/pipelined systolic array for weight extraction for RLS and CRLS adaptive array systems without the need of the backsolve is described. There are two procedures required in both systems. First, for a parallel/pipelined RLS weight extraction system, the initialization procedure computes the initial upper triangular matrix $R(N)$, a vector $\underline{u}(N)$, and a lower triangular matrix $R^{-H}(N)$ for time $0 \leq n \leq N$ where N is the number of sensors. Second, the recursive procedure is employed to update the whole system and to compute the optimal weight vector recursively. For the parallel/pipelined CRLS weight extraction algorithm, the initialization procedure computes the initial upper triangular matrix $R(N)$, a parameter vector $\underline{z}(N)$, and a lower triangular matrix $R^{-H}(N)$. The re-

cursive procedure is used to update the system and to compute the optimal weight vector for CRLS adaptive beamforming. It is well-known [2] that the solution of the parallel/pipelined weight extraction problem is defined only in the recursive procedure during time $n > N$. The proposed RLS and CRLS adaptive array systems have four advantages: (1) there is no bottleneck in the whole architecture since QR decomposition by the square root free fast Givens method is used, (2) the whole architecture has a fully pipelined design since backward substitution is avoided, (3) it is a single fully pipelined open-loop system without any feedback arrangement, and (4) the parallel/pipelined weight extraction system functions recursively to update the instantaneous optimal weight vector since only one mode is required in the recursive procedure. Therefore, the new VLSI systolic architectures achieve minimal memory and maximal parallelism for real-time signal processing applications and VLSI hardware implementation. A similar architectural concept without mathematics details for updating optimal weights presented independently by Shepherd, McWhirter, and Hudson recently [10]. Their proposed architecture reinforces our work in this chapter.

This chapter is organized as follows: In Section 2, the background and the new techniques implementing the RLS and CRLS adaptive array systems into systolic arrays structures are introduced. The problem of arranging an RLS adaptive array system into a systolic array structure is presented in Sections 3

to 6. In Section 3 the recursive least squares (RLS) array problem is discussed. Then in sections 4 and 5, a parallel/pipelined RLS weight extraction algorithm with an initialization procedure and recursive procedure and their systolic array processors which do not need backsolving are described. In Section 6 the fast Givens method is employed for RLS array systems. This has the advantage of the speeding up the systems. Another important adaptive array system, the constrained recursive least squares (CRLS) system, is presented in Section 7. In Sections 8 and 9, the parallel/pipelined CRLS weight extraction algorithm is presented without the need for a backsolving algorithm and systolic array processors are illustrated. In the last section, the fast Givens method is used for a CRLS adaptive array system to speed it up.

3.2 RLS Adaptive Array System

In this section we address the partially adaptive beamforming problem by considering the recursive least squares (RLS) method for choosing statistically optimal weights to generate the output residual. The block diagram of a partially adaptive beamforming system such as a sidelobe noise cancellation system is shown in Figure 3.4.

The sidelobe cancellation technique is employed to suppress the sidelobe interference and noise by subtracting the estimate from the radar main channel output. It is easy to see from Figure 3.4 that the output at *ith* snapshot can

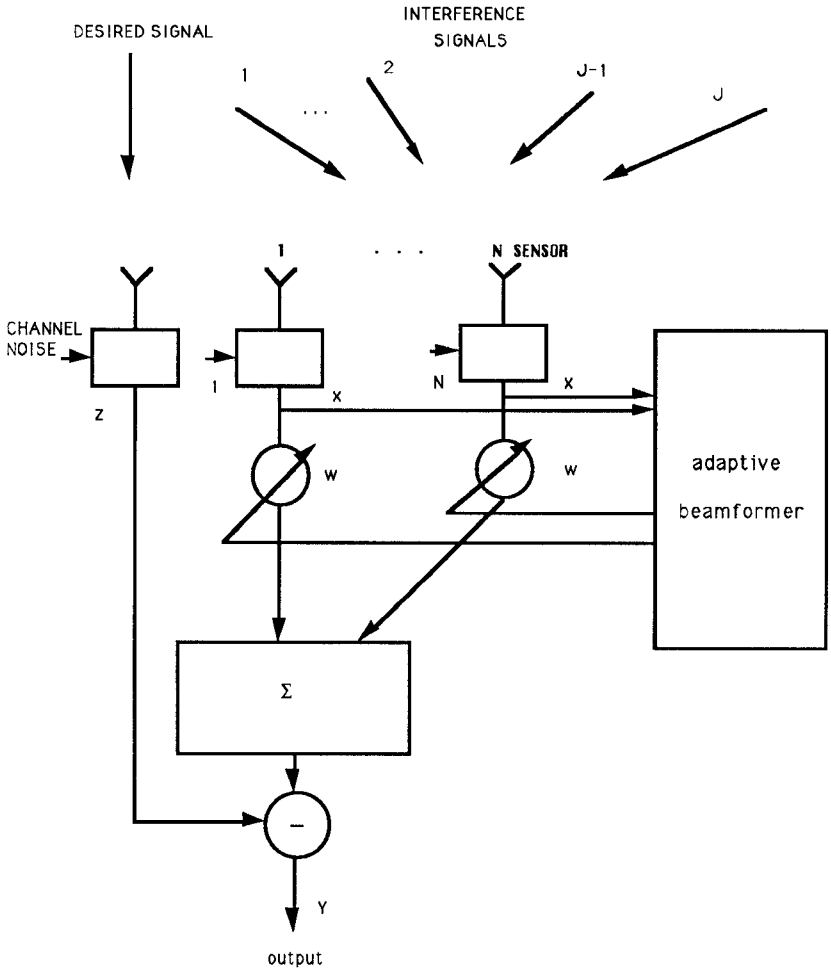


Figure 3.4: Sidelobe Cancellation Adaptive Beamforming System

be expressed as

$$y(i) = \sum_{l=1}^{N} x_l(i) w_l - z(i) \tag{3.5}$$

Equation 3.5 can be represented in the matrix form.

$$\underline{y}(n) = X(n)\underline{w}(n) - \underline{z}(n) \tag{3.6}$$

where $\underline{y}(n)$ is an n by 1 output vector matrix

$$\underline{y}(n) = \begin{bmatrix} y(t_1) \\ y(t_2) \\ \vdots \\ y(t_n) \end{bmatrix}, \tag{3.7}$$

$\underline{z}(n)$ is an n by 1 desired input data vector matrix

$$\underline{z}(n) = \begin{bmatrix} z(t_1) \\ z(t_2) \\ \vdots \\ z(t_n) \end{bmatrix}, \tag{3.8}$$

$X(n)$ is an n by N observed input data matrix

$$X(n) = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_N(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_N(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_n) & x_2(t_n) & \cdots & x_N(t_n) \end{bmatrix}, \tag{3.9}$$

an $\underline{w}(n)$ is an N by 1 weight vector

$$\underline{w}(n) = \begin{bmatrix} w_1(t_n) \\ w_2(t_n) \\ \vdots \\ w_N(t_n) \end{bmatrix}. \tag{3.10}$$

The aim of the optimal adaptive array system is to minimize the output residual of the total interferences and noises by adjusting the weights while maintaining a fixed gain in the direction of the desired signal; i.e. to minimize the sum of the squares of the elements of the n by 1 output residual vector $\underline{y}(n)$. This leads to a maximization of output residual signal to noise (including interferences and receiver noises) ratio. Therefore, we have the least-squares problem

$$\min_{\underline{w}} \ ||\underline{y}(n)||_2 = \min_{\underline{w}} \ ||X(n)\underline{w}(n) - \underline{z}(n)||_2 \tag{3.11}$$

The solution to this minimization problem is

$$\underline{w}_{LS}(n) = (X^H(n)X(n))^{-1}X^H(n)\underline{z}(n) \tag{3.12}$$

This is called the normal equation.

This solution using the normal equation, generally known as the sample matrix inversion (SMI) method in adaptive array and multichannel applications, may sometimes be sensitive to roundoff errors and can not be implemented by systolic arrays for VLSI hardware implementation. Therefore, the major

issues in this chapter are: (1) to compute the optimal weight vector based on the QR decomposition for improved numerical accuracy and (2) to map the QR-based weight extraction algorithm into a systolic array processors which does not need backward substitution.

3.3 QR Based RLS Algorithm

The least squares problem introduced may be solved by the numerically stable QR decomposition which will now be described [6,11]. Applying the QR decomposition to the data $X(n)$, we have

$$Q(n)X(n) = \begin{bmatrix} R(n) \\ \\ 0 \end{bmatrix} \tag{3.13}$$

where $R(n)$ is an $N \times N$ upper triangular matrix and $Q(n)$ is a unitary matrix which has the property $Q^H(n)Q(n) = I$.

Applying the same unitary matrix $Q(n)$ to the desired data $\underline{z}(n)$ gives

$$Q(n)\underline{z} = \begin{bmatrix} \underline{u}(n) \\ \\ \underline{v}(n) \end{bmatrix} \tag{3.14}$$

where $\underline{u}(n)$ is an $N \times 1$ vector and $\underline{v}(n)$ is an $(n - N) \times 1$ vector. The optimal weight vector is obtained by substituting Equations 3.13 and 3.14 into Equation 3.12. The resulting QR-based weight vector is

$$\underline{w}_{LS}(n) = R^{-1}(n)\underline{u}(n) \tag{3.15}$$

3.3.1 Initialization Procedure

The initial upper triangular matrix $R(N)$ given by applying QR decomposition to the data $X(N)$ has the form

$$R(N) = Q(N)X(N) \tag{3.16}$$

Then by applying the same $Q(N)$ to the desired data $\underline{z}(N)$, the initial data vector of $\underline{u}(N)$ is given by

$$\underline{u}(N) = Q(N)\underline{z}(N) \tag{3.17}$$

Combining Equations 3.16 and 3.17, we obtain

$$\begin{bmatrix} R(N) & \underline{u}(N) \end{bmatrix} = Q(N) \begin{bmatrix} X(N) & \underline{z}(N) \end{bmatrix} \tag{3.18}$$

Finally, the initial lower triangular matrix $R^{-H}(N)$ is computed by using the systolic forward substitution algorithm described in 2.4 which is

$$R^{-H}(N) = R^{-H}(N)I \tag{3.19}$$

3.3.2 Recursive Updating

In this subsection the parallel/ pipelined RLS weight extraction system is computed recursively to update the instantaneous optimal weight vector for each new data sample vector. It is known [24] that the unitary matrix $\widehat{Q}(n)$

can be applied to update the upper triangular matrix $R(n-1)$ as follows

$$\widehat{Q}(n)\begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix} \tag{3.20}$$

The same unitary matrix $\widehat{Q}(n)$ is used to update $\underline{u}(n-1)$ by the equation

$$\widehat{Q}^H(n)\begin{bmatrix} \beta\underline{u}(n-1) \\ \beta\underline{v}(n-1) \\ y(t_n) \end{bmatrix} = \begin{bmatrix} \underline{u}(n) \\ \beta\underline{v}(n-1) \\ \alpha(n) \end{bmatrix} \tag{3.21}$$

It is necessary to update the lower triangular matrix $R^{-H}(n-1)$ for computing the instantaneous optimal RLS weighting vector. It is also known [9] that the same unitary matrix $\widehat{Q}(n)$, which updates the upper triangular matrix $R(n-1)$, can also be used to update the lower triangular matrix $R^{-H}(n-1)$. The reason is explained by following equation.

$$\begin{aligned} I &= R^H(n)R^{-H}(n) \\ &= R^H(n-1)R^{-H}(n-1) \\ &= \begin{bmatrix} \beta R^H(n-1) & 0 & \underline{x}^*(t_n) \end{bmatrix} \widehat{Q}^H(n)\widehat{Q}(n)\begin{bmatrix} \frac{1}{\beta}R^{-H}(n-1) \\ \# \\ 0 \end{bmatrix} \end{aligned} \tag{3.22}$$

where # denotes an arbitrary vector of no interest in mathematical and physical concept. Therefore,

$$\widehat{Q}(n)\begin{bmatrix} \frac{1}{\beta}R^{-H}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} R^{-H}(n) \\ \# \\ \# \end{bmatrix} \tag{3.23}$$

Let us summarize the updating procedure for the upper triangular matrix $R(n-1)$, the vector $\underline{u}(n-1)$, and the lower triangular matrix $R^{-H}(n-1)$ at same time.

$$\begin{aligned} \widehat{Q}(n)\begin{bmatrix} \beta R(n-1) & \vdots & \beta\underline{u}(n-1) & \vdots & \frac{1}{\beta}R^{-H}(n-1) \\ 0 & \vdots & \beta\underline{v}(n-1) & \vdots & \# \\ \underline{x}^T(t_n) & \vdots & z(t_n) & \vdots & 0 \end{bmatrix} \\ = \begin{bmatrix} R(n) & \vdots & \underline{u}(n) & \vdots & R^{-H}(n) \\ 0 & \vdots & \beta\underline{v}(n-1) & \vdots & \# \\ 0 & \vdots & \alpha & \vdots & \# \end{bmatrix} \end{aligned} \tag{3.24}$$

Therefore, the weight vector for RLS problem is given by

$$\underline{w}_{RLS}(n) = R^{-1}(n)\underline{u}(n) \tag{3.25}$$

3.4 Systolic RLS Weight Extraction System

In this section we first summarize the systolic RLS weight extraction algorithm shown in Table 3.1. Then, a single fully pipelined structure is illustrated for

the case of four sensors to receive the observed data and a desired data and is shown in Figure 3.5 [10]. Our systolic RLS architecture consists of four processor elements which are shown in Figure 3.6. The mode 1 algorithm for the RLS systolic array for each processor element is described in Table 3.2 and the mode 2 algorithm is presented in Table 3.3. To operate the RLS systolic array processors, the initialization procedure is used for time $0 \leq n \leq N$ to obtain the initial upper triangular matrix $R(N)$, the initial parameter vector $\underline{u}(N)$, and the lower triangular matrix $R^{-H}(N)$. It is shown in Figure 3.5 that the two different modes described in Table 3.2 and 3.3 are used in the initialization procedure. The recursive procedure for parallel/pipelined weight extraction used at times $n > N$ only requires mode 1. Since the optimal weight vector only occurs in the recursive procedure, the parallel/pipelined RLS weight extraction system described is very promising for VLSI hardware implementation and real-time signal processing applications.

3.5 Fast Givens Based RLS Algorithm

As mentioned in Section 2, QR decomposition by the fast Givens method is carried out by a diagonal matrix and an upper triangular matrix as follows.

$$R(n) = D^{\frac{1}{2}}(n)\overline{R}(n) \tag{3.26}$$

$$\underline{u}^{i}(n) = D^{\frac{1}{2}}\underline{\overline{u}}(n) \tag{3.27}$$

1. Initialize Conditions at $n = 0$ by setting

   $R(0) = 0 \quad \underline{z}(0) = 0 \quad R^{-H}(0) = 0$

2. Initialization Procedure for $0 \leq n \leq N$:

   (a) $Q(N) \left[\begin{array}{c:c} X(N) & \underline{z}(N) \end{array} \right] = \left[\begin{array}{c:c} R(N) & \underline{u}(N) \end{array} \right]$ (Mode 1)

   (b) $R^{-H}(N) = R^{-H}(N)I$ (Mode 2)

3. Recursive Procedure for $n > N$ (Mode 1 only):

   (a) $\widehat{Q}(n) \begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix}$

   (b) $\widehat{Q}(n) \begin{bmatrix} \beta \underline{u}(n-1) \\ \beta \underline{v}(n-1) \\ z(t_n) \end{bmatrix} = \begin{bmatrix} \underline{u}(n) \\ \beta \underline{v} \\ \alpha \end{bmatrix}$

   (c) $\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta} R^{-H}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} R^{-H}(n) \\ \# \\ \# \end{bmatrix}$

   (d) $\underline{w}_{RLS}(n) = R^{-1}(n)\underline{u}(n)$

Table 3.1: Summary of Parallel/Pipelined QRD-RLS Algorithm

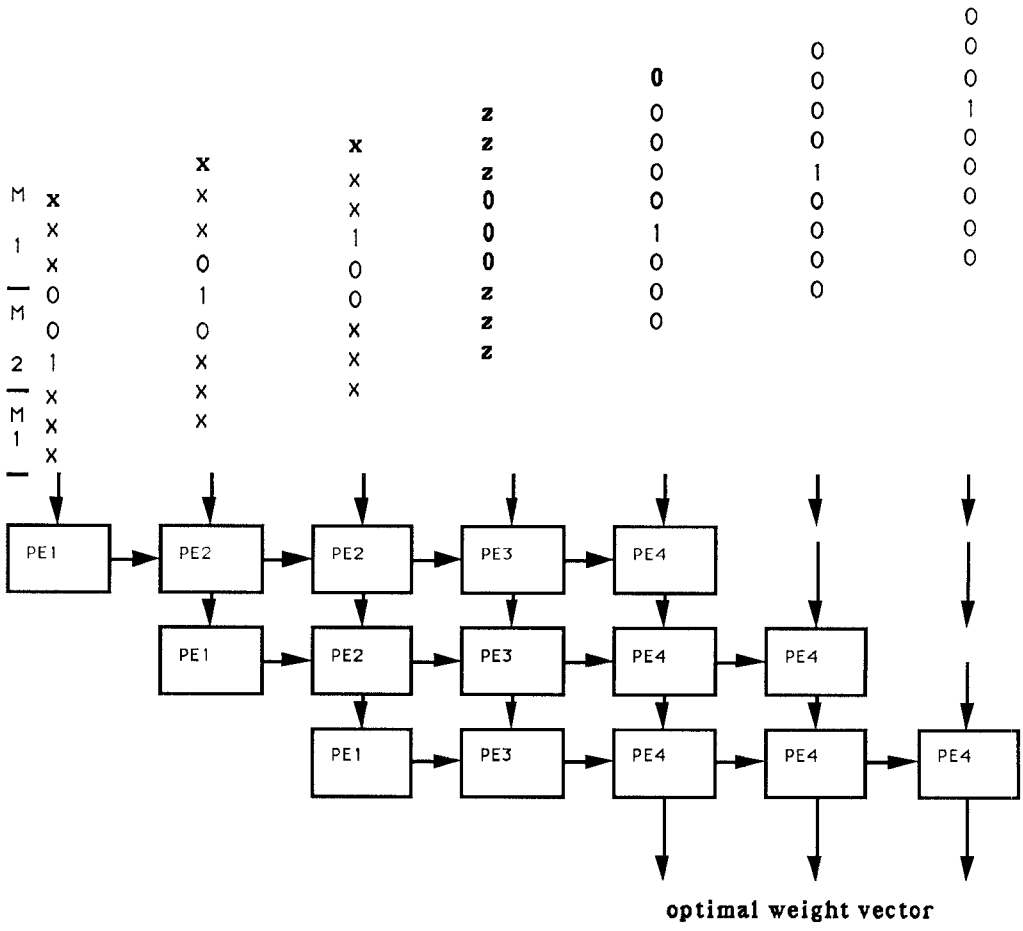


Figure 3.5: RLS Systolic Array Processors [10]

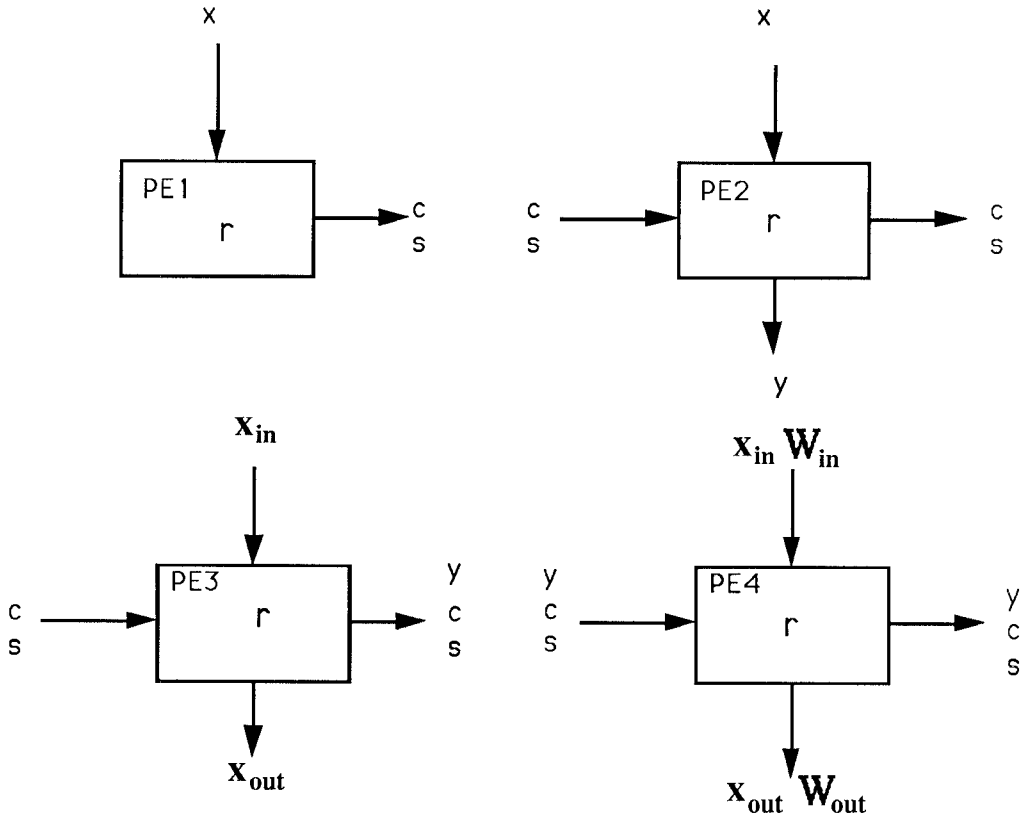


Figure 3.6: Processor Elements of RLS Systolic Array Processor

$PE1$	$PE2$	$PE3$	$PE4$
$d \leftarrow (\beta^2 r^2 + \lvert x \rvert^2)^{\frac{1}{2}}$	$y \leftarrow -s\beta r + cx$	$x_{out} \leftarrow -\frac{1}{\beta} sr + cx_{in}$	$x_{out} \leftarrow -\frac{1}{\beta} sr + cx_{in}$
$c \leftarrow \frac{\beta r}{d}$	$r \leftarrow c\beta r + s^* x$	$r \leftarrow \frac{1}{\beta} cr + s^* x_{in}$	$r \leftarrow \frac{1}{\beta} cr + s^* x_{in}$
$s \leftarrow \frac{x}{d}$		$y \leftarrow r$	$w_{out} \leftarrow yr^* + w_{in}$
$r \leftarrow d$			

Table 3.2: The Givens Based-RLS Algorithm of Mode 1

$PE1$	$PE2$	$PE3$	$PE4$
$s \leftarrow \frac{x}{r}$	$y \leftarrow cx - sr$	$y \leftarrow cx - sr$	$x_{out} \leftarrow x_{in}$
$c \leftarrow 1$			$if \;\; x_{in} \leftarrow 1$
			$then \;\; r \leftarrow s*$

Table 3.3: The Givens Based-RLS Algorithm of Mode 2

$$R^{-H}(n) = D^{\frac{-1}{2}}\overline{R}^{-H}(n) \tag{3.28}$$

According to the RLS algorithm based on fast Givens method, in the initialization procedure, a diagonal matrix $D^{\frac{1}{2}}(N)$, an upper triangular matrix $\overline{R}(N)$, and a vector $\underline{\overline{u}}(N)$ are generated in the same mode.

$$Q(N)\left[\begin{array}{c:c} X(N) & \underline{y}(N) \end{array}\right] = D^{\frac{1}{2}}(N)\left[\begin{array}{c:c} \overline{R}(N) & \underline{\overline{u}}(N) \end{array}\right] \tag{3.29}$$

By using the other mode, the initial lower triangular matrix $\overline{R}^{-H}(N)$ is then computed as follows.

$$\overline{R}^{-H}(N) = \overline{R}^{-H}(N)I \tag{3.30}$$

In order to update the optimal weight vector recursively, a vector $\underline{\overline{u}}(n-1)$ and a lower triangular matrix $R^{-H}(n-1)$ must be updated at each new data sample. The following equation shows how to update the whole system together.

$$\begin{aligned}\widehat{Q}(n)&\left[\begin{array}{c:c:c} \beta D^{\frac{1}{2}}\overline{R}(n-1) & \beta\underline{\overline{u}}(n-1) & \frac{1}{\beta}\overline{R}^{-H}(n-1) \\ 0 & \beta\underline{\overline{v}}(n-1) & \# \\ \underline{x}^T(t_n) & y(t_n) & 0 \end{array}\right] \\ &= \left[\begin{array}{c:c:c} D^{\frac{1}{2}}(n)\overline{R}(n) & \underline{\overline{u}}(n) & \overline{R}^{-H}(n) \\ 0 & \beta\underline{\overline{v}}(n-1) & \# \\ 0 & \# & \# \end{array}\right]\end{aligned} \tag{3.31}$$

where # denotes an arbitrary matrix or vector with no special interest.

Finally, the optimal weight vector, which can be updated recursively, has

the form

$$\underline{w}(n) = \overline{R}^{-1}(n)\overline{\underline{u}}(n) \tag{3.32}$$

The algorithm and systolic array processors for the fast Givens-RLS algorithm with modifications from those of Givens-RLS algorithm are described in Table 3.4 and Figure 3.7. The proposed fast RLS weight extraction system consists of four processor elements with two modes. The symbols of these four processor elements are illustrated in Figure 3.8 and the functions of each processor element in the two modes are also described in Tables 3.5 and 3.6.

1. Initialize Conditions at $n = 0$ by setting

   item $D^{\frac{1}{2}}(0)\overline{R}(0) = 0 \quad \underline{\overline{u}}(0) = 0 \quad \overline{R}^{-H}(0) = 0$

2. Initialization Procedure for $0 \leq n \leq N$:

   (a) $Q(N)\left[\begin{array}{c:c} X(N) & \underline{z}(N) \end{array}\right] = D^{\frac{1}{2}}(N)\left[\begin{array}{c:c} \overline{R}(N) & \underline{\overline{u}}(N) \end{array}\right]$

   (Mode 1)

   (b) $\overline{R}^{-H}(N) = \overline{R}^{-H}(N)I$ (Mode 2)

3. Recursive Procedure for $n > N$ (Mode 1 only):

   (a) $\widehat{Q}(n)\begin{bmatrix} \beta D^{\frac{1}{2}}(n-1)\overline{R}(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} D^{\frac{1}{2}}(n)\overline{R}(n) \\ 0 \\ 0 \end{bmatrix}$

   (b) $\widehat{Q}(n)\begin{bmatrix} \beta D^{\frac{1}{2}}(n-1)\underline{\overline{u}}(n-1) \\ \# \\ z(t_n) \end{bmatrix} = \begin{bmatrix} D^{\frac{1}{2}}(n)\underline{\overline{u}}(n) \\ \# \\ \# \end{bmatrix}$

   (c) $\overline{R}^{-H}(n) = \widehat{Q}(n)\begin{bmatrix} \frac{1}{\beta}\overline{R}^{-H}(n-1) \\ \# \\ 0 \end{bmatrix}$

   (d) $\underline{w}(n) = \overline{R}^{-1}(n)\underline{\overline{u}}(n)$

Table 3.4: Summary of Fast Givens-RLS Algorithm

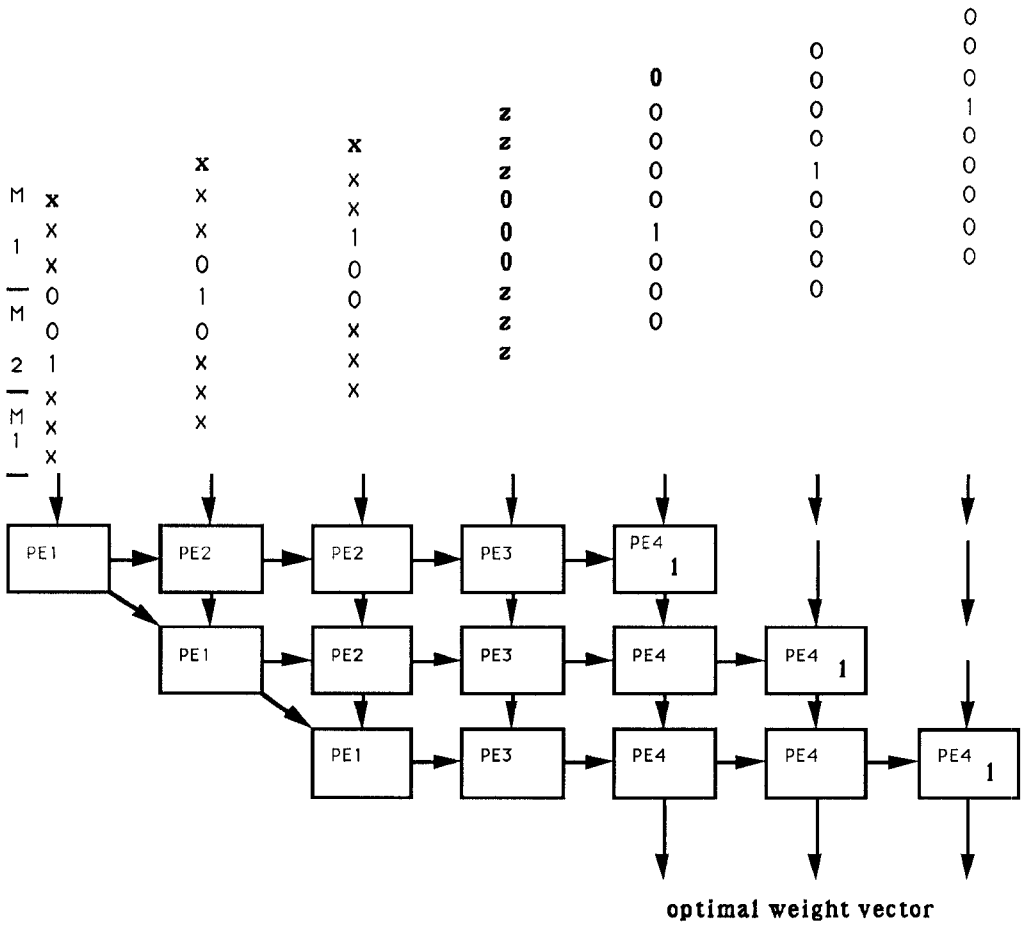


Figure 3.7: Fast RLS Systolic Array Processor

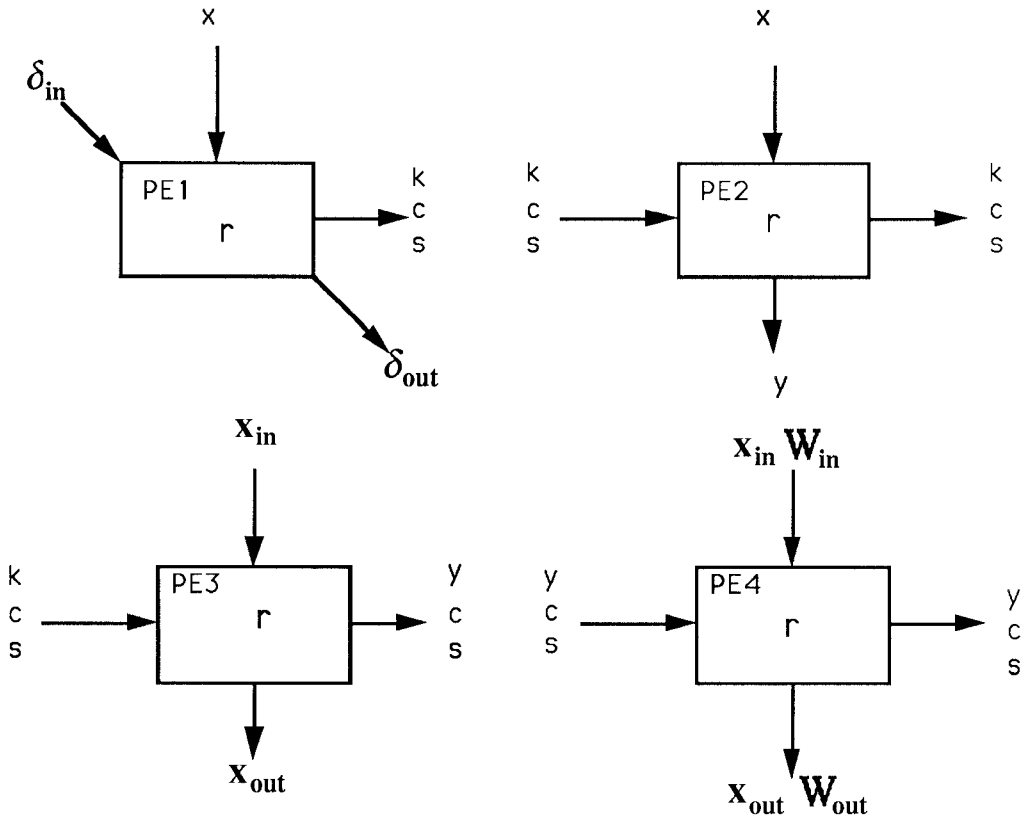


Figure 3.8: Processor Elements of Fast RLS Systolic Array Processor

$PE1$	$PE2$	$PE3$	$PE4$
$k \leftarrow \beta^2 r + \delta_{in}\lvert x\rvert^2$	$y \leftarrow -c\beta r + x$	$x_{out} \leftarrow -c\beta r + x_{in}$	$x_{out} \leftarrow -\frac{1}{\beta}cr + x_{in}$
$\delta_{out} \leftarrow \frac{\beta^2 r \delta_{in}}{k}$	$r \leftarrow \beta r + s^* y$	$r \leftarrow \beta r + s^* x_{out}$	$r \leftarrow \frac{1}{\beta} r + s^* x_{out}$
$c \leftarrow x$		$y \leftarrow \frac{r}{k}$	$w_{out} \leftarrow y r^* + w_{in}$
$s \leftarrow \frac{\delta_{in} x}{k}$			
$r \leftarrow k$			

Table 3.5: The Fast Givens-RLS Algorithm of Mode 1

$PE1$	$PE2$	$PE3$	$PE4$
$s \leftarrow x$	$y \leftarrow cx - sr$	$x_{out} \leftarrow x_{in}$	$x_{out} \leftarrow x_{in}$
$c \leftarrow 1$			$if \quad x_{in} \leftarrow 1$
			$then \quad r \leftarrow s*$

Table 3.6: The Fast Givens-RLS Algorithm of Mode 2

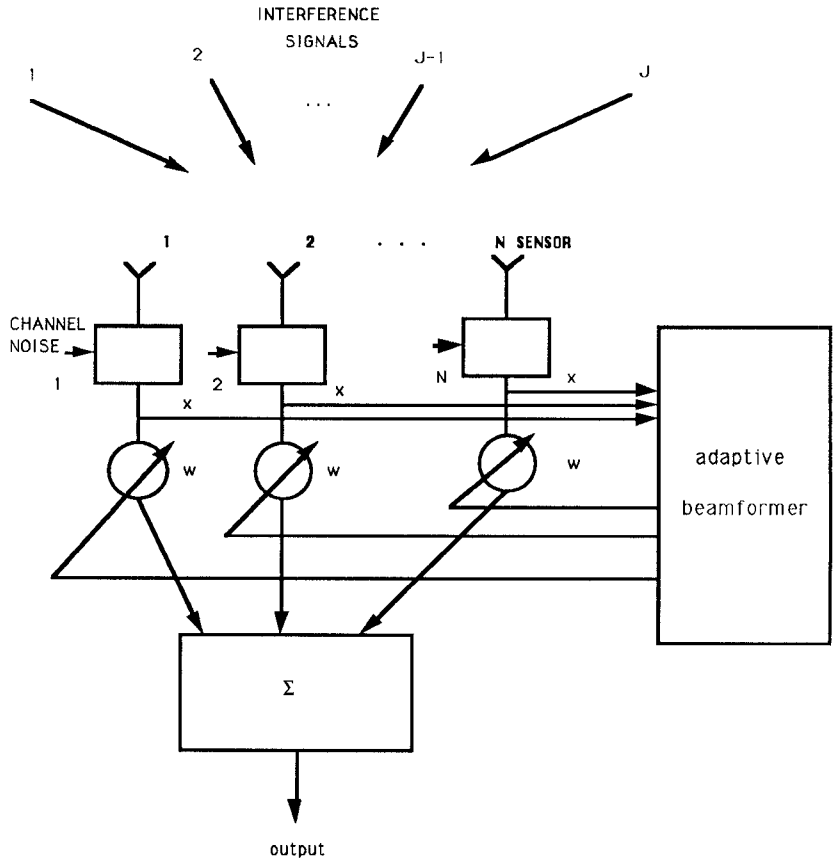


Figure 3.9: CRLS Adaptive Beamforming System

3.6 CRLS Adaptive Array System

In this section, we consider a constrained recursive least squares (CRLS) problem for a fully adaptive beamforming system. The block diagram of a general adaptive beamforming system is shown in Figure 3.9. An adaptive beamformer is a processor used in conjunction with an array of sensors to adjust its weights based on a certain criterion so as to provide versatile beam patterns

to suppress the interference signals. From Figure 3.9 an output y at the kth snapshot of the adaptive beamformer is expressed as a linear combination of weighted inputs,

$$y_k = \sum_{l=1}^{N} x_{k,l} w_l \tag{3.33}$$

Furthermore, if n snapshots of input data flow into the system, the output of the adaptive beamformer is

$$\underline{y}(n) = X(n)\underline{w}(n) \tag{3.34}$$

where $X(n)$ is an n by N input matrix which consists of n row vectors. Each row vector results from a snapshot by N sensors, so

$$X(n) = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_N(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_N(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_n) & x_2(t_n) & \cdots & x_N(t_n) \end{bmatrix}, \tag{3.35}$$

$\underline{w}(n)$ is an N by 1 weight vector given by

$$\underline{w}^T(n) = \begin{bmatrix} w_1(t_n) & w_2(t_n) \cdots w_N(t_n) \end{bmatrix}, \tag{3.36}$$

$\underline{y}(n)$ is an n by 1 output vector given by

$$\underline{y}^T(n) = \begin{bmatrix} y(t_1) & y(t_2) & \cdots & y(t_n) \end{bmatrix}, \tag{3.37}$$

and the superscript T denotes transpose. A signal received from the direction of interest θ_i is called a beamformer response r^i given by

$$r^i = \underline{c}^{i^H} \underline{w} \tag{3.38}$$

where $\underline{c}^{iH}$ is the N by 1 mainbeam steering direction vector given by

$$\underline{c}^{iH} = \begin{bmatrix} 1 & e^{j\frac{2\pi}{\lambda}Dsin(\theta_i)} & e^{j\frac{2\pi}{\lambda}2\,Dsin(\theta_i)} & \cdots & e^{j\frac{2\pi}{\lambda}(N-1)\,Dsin(\theta_i)} \end{bmatrix}$$

where D is the distance between two sensors in the array, λ is the wavelength, and the superscript H denotes Hermitian. Taking expectation of $\underline{y}^H(n)\underline{y}(n)$ yields

$$E[\underline{y}^H(n)\underline{y}(n)] = E[\underline{w}^H(n)X^H(n)X(n)\underline{w}(n)] = \underline{w}^H(n)M(n)\underline{w}(n) \qquad (3.39)$$

where $M(n)$ is the n by n covariance matrix for $X(n)$.

In general, to design a fully adaptive array beamforming system is to solve a constrained optimization problem where a beamformer response is constrained by a fixed gain. Then the CRLS problem is

$$\min_{\underline{w}} \ \underline{w}^H M \underline{w}$$

$$\text{subject to } \underline{c}^{iH}\underline{w} = r^i \quad for \ \ i = 1, 2, \cdots, K.$$

Using the method of Lagrange multipliers, the optimal weight vector, $\underline{w}^i_{opt}$, is found to be

$$\underline{w}^i_{opt}(n) = \frac{r^i M^{-1}(n)\underline{c}^i}{\underline{c}^{iH}M^{-1}(n)\underline{c}^i} \quad for \ \ i = 1, \cdots, K \qquad (3.40)$$

In practice, $M(n)$ used in Equation 3.40 is the sample covariance matrix of the observed data $X(n)$,

$$M(n) = X^H(n)X(n) \qquad (3.41)$$

not the covariance matrix of $X(n)$ used in Equation 3.39.

Pipelined data-parallel algorithms that can be implemented on systolic array processors (SAPs) are called *systolic algorithms* (SAs). The SAs designed in this thesis not only can be implemented by SAPs but also are numerically stable, an important property which the Sample Matrix Inversion method does not possess. SAPs have additional nice properties such as simplicity, modularity, and expandability which are very attractive for implementations of CRLS adaptive beamforming systems. As shown in Equation 3.40, adaptive beamforming involves a procedure for inverting a sample covariance matrix. It is known that a commonly used method, the sample matrix inversion (SMI) method, generally leads to some undesirable numerical characteristics if the sample covariance matrix is ill-conditioned. In order to alleviate this difficulty, the QR decomposition can be used. In the rest of this chapter, we introduce a systolic CRLS algorithm based on a QR decomposition approach. Our algorithms will be designed by using the Givens method and fast Givens method.

3.7 QR Based CRLS Algorithm

In this section, a numerically stable and fully pipelined algorithm is introduced for weight extraction in a systolic CRLS adaptive array system. A QR

decomposition is applied to the input observed data $X(n)$, so that

$$Q(n)X(n) = \begin{bmatrix} R(n) \\ \\ 0 \end{bmatrix} \tag{3.42}$$

Substituting Equation 3.42 into 3.41, we have

$$M(n) = R^H(n)R(n) \tag{3.43}$$

On substituting Equation 3.43 into 3.40, the CRLS adaptive weight vector becomes

$$\underline{w}^i_{CRLS} = \frac{r^i R^{-1}(n) R^{-H}(n) \underline{c}^i}{\underline{c}^{iH} R^{-1}(n) R^{-H}(n) \underline{c}^i} \quad for \;\; i = 1, \cdots, K. \tag{3.44}$$

where the lower triangular matrix R^{-H} is equal to $(R^H)^{-1}$.

Define a parameter vector $\underline{z}(n)$, so that

$$\underline{z}^i(n) = R^{-H}(n)\underline{c}^i \tag{3.45}$$

The weight vector of CRLS adaptive array system is given by

$$\underline{w}^i_{CRLS} = \frac{r^i}{|\underline{z}^i|^2} R^{-1}(n)\underline{z}^i(n) \quad for \;\; i = 1, \cdots, K. \tag{3.46}$$

3.7.1 Initialization Procedure

By applying the QR decomposition to the observed data $X(N)$, the initial upper triangular matrix $R(N)$ is given by

$$R(N) = Q(N)X(N) \tag{3.47}$$

where $X(N)$ is an N by N observed data matrix consisting of N row vectors and each row vector is a snapshot of N sensors, and Q is an N by N unitary matrix which satisfies the property of $Q^H Q = I$. Then the initial parameter vector $\underline{z}^i(N)$ can be computed by using the systolic forward substitution (FS) algorithm

$$\underline{z}^i(N) = R^{-H}(N)\underline{c}^i \tag{3.48}$$

Finally, the initial lower triangular matrix $R^{-H}(N)$ is obtained by replacing the steering vector in Equation 3.48 by an identity matrix, so that

$$R^{-H}(N) = R^{-H}(N)I \tag{3.49}$$

Combining the Equations 3.48 and 3.49 gives

$$\left[\begin{array}{c:c} \underline{z}^i(N) & R^{-H}(N) \end{array} \right] = R^{-H}(N) \left[\begin{array}{c:c} \underline{c}^i & I \end{array} \right] \tag{3.50}$$

3.7.2 Recursive Updating

It has been shown in [14] that a QR decomposition of $X(n)$ can be carried out recursively. To update the upper triangular matrix $R(n-1)$, a unitary matrix $\widehat{Q}(n)$ is used with form

$$\widehat{Q}(n) \begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix} \tag{3.51}$$

where β, the forgetting factor, is defined as the weight factor applied to the previous data. To update a parameter vector $\underline{z}(n-1)$, notice that

$$\begin{aligned}
\underline{c}^i &= R^H(n)\underline{z}(n) \\
&= R^H(n-1)\underline{z}(n-1) \\
&= \begin{bmatrix} \beta R^H(n-1) & 0 & \underline{x}^*(t_n) \end{bmatrix} \widehat{Q}^H(n)\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{z}(n-1) \\ \# \\ 0 \end{bmatrix}
\end{aligned} \tag{3.52}$$

Therefore, we have

$$\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{z}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{z}(n) \\ \# \\ \# \end{bmatrix} \tag{3.53}$$

where # denotes an arbitrary vector of no interest in mathematical and physical concept.

As described in Section 3.3.2, the same unitary matrix used to update the upper triangular matrix $R(n-1)$ can be employed to update the lower triangular R^{-H}. Therefore,

$$\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}R^{-H}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} R^{-H}(n) \\ \# \\ \# \end{bmatrix} \tag{3.54}$$

Let us summarize the updating procedure for an upper triangular matrix $R(n-$

1), a vector $\underline{z}(n-1)$, and a lower triangular $R^{-H}(n-1)$ at same time.

$$\widehat{Q}(n)\begin{bmatrix} \beta R(n-1) & \vdots & \frac{1}{\beta}\underline{z}(n-1) & \vdots & \frac{1}{\beta}R^{-H}(n-1) \\ 0 & \vdots & \# & \vdots & \# \\ \underline{x}^T(t_n) & \vdots & 0 & \vdots & 0 \end{bmatrix} = \begin{bmatrix} R(n) & \vdots & \underline{z}^i(n) & \vdots & R^{-H}(n) \\ 0 & \vdots & \# & \vdots & \# \\ 0 & \vdots & \# & \vdots & \# \end{bmatrix} \tag{3.55}$$

For convenience, let us define $\underline{\widehat{w}}^i(n)$ as

$$\underline{\widehat{w}}^i(n) = R^{-1}(n)\underline{z}^i(n) \tag{3.56}$$

It is straightforward to obtain the $\underline{\widehat{w}}^i(n)$ by using the updated $\underline{z}^i(n)$ and the updated $R^{-H}(n)$ in the systolic arrays. As a result, the weight vector $\underline{\widehat{w}}^i(n)$ is also updated. Finally, the weight vector for the CRLS adaptive array system is given by

$$\underline{w}^i_{CRLS}(n) = \frac{r^i}{|\underline{z}^i(n)|^2}\underline{\widehat{w}}^i(n) \quad for \;\; i = 1, \cdots, K. \tag{3.57}$$

3.8 Systolic CRLS Weight Extraction System

A single fully pipelined structure is shown for the case of three sensors and one constraint in Figure 3.10. Another single fully pipelined structure is shown for three sensors and two constraints in Figure 3.11. There are five processor elements shown in Figure 3.12 for our proposed fully pipelined structure.

1. Initialize Conditions at $n = 0$ by setting

   $R(0) = 0 \quad \underline{z}(0) = 0 \quad R^{-H}(0) = 0$

2. Initialization Procedure for $0 \leq n \leq N$:

   (a) $Q(N)X(N) = R(N)$ (Mode 1)

   (b) $\left[\underline{z}^i(N) \ \vdots \ R^{-H}(N) \right] = R^{-H}(N) \left[\underline{c}^i \ \vdots \ I \right]$ (Mode 2)

3. Recursive Procedure for $n > N$ (Mode 1 only):

   (a) $\widehat{Q}(n) \begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix}$

   (b) $\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{z}^i(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{z}^i(n) \\ \# \\ \# \end{bmatrix}$

   (c) $\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}R^{-H}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} R^{-H}(n) \\ \# \\ \# \end{bmatrix}$

   (d) $\underline{w}(n) = \frac{r^i}{|\underline{z}^i(n)|^2} R^{-1}(n)\underline{z}^i(n)$

Table 3.7: Summary of Parallel/Pipelined QRD-CRLS Algorithm

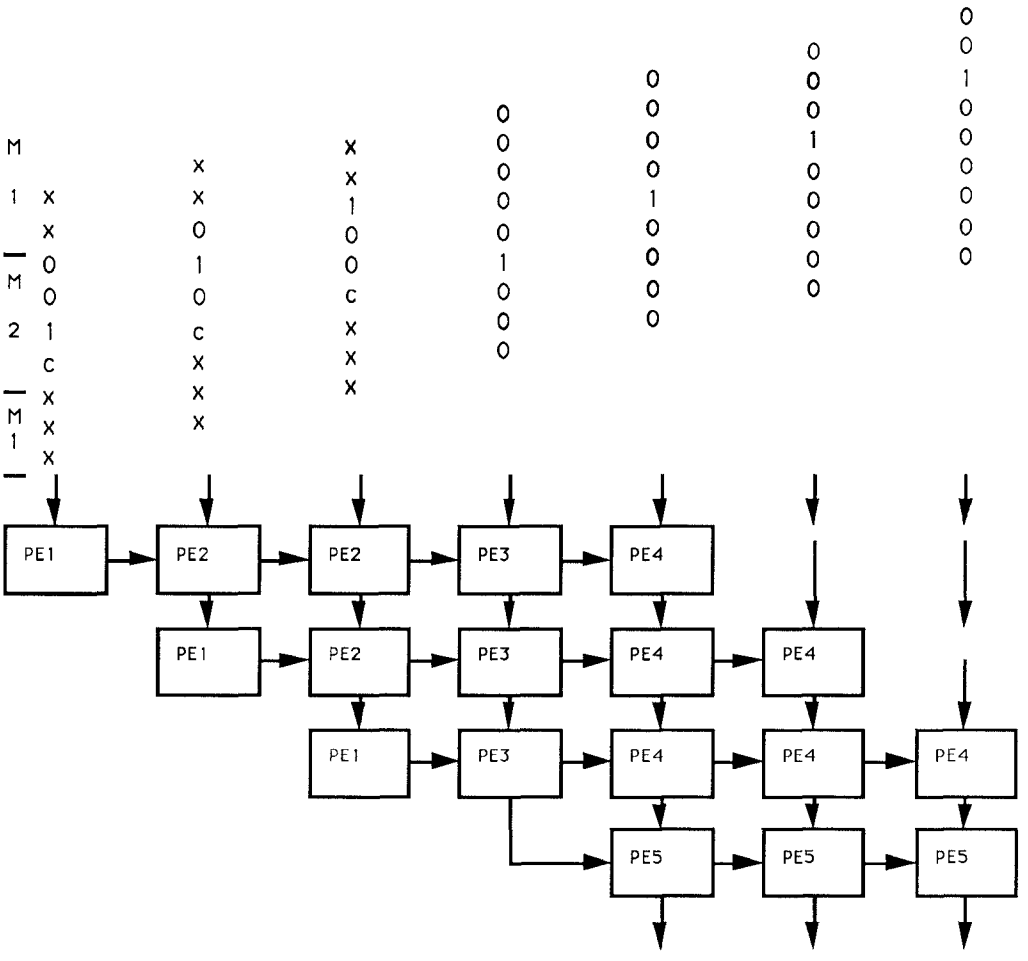


Figure 3.10: CRLS Systolic Array Processors (Case 1) [4,10]

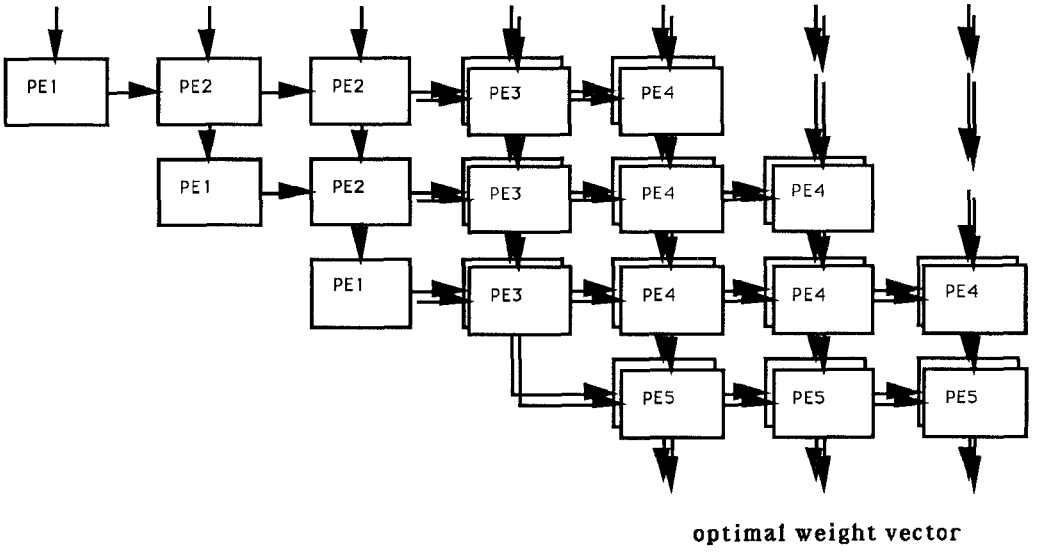


Figure 3.11: CRLS Systolic Array Processors (Case 2) [4]

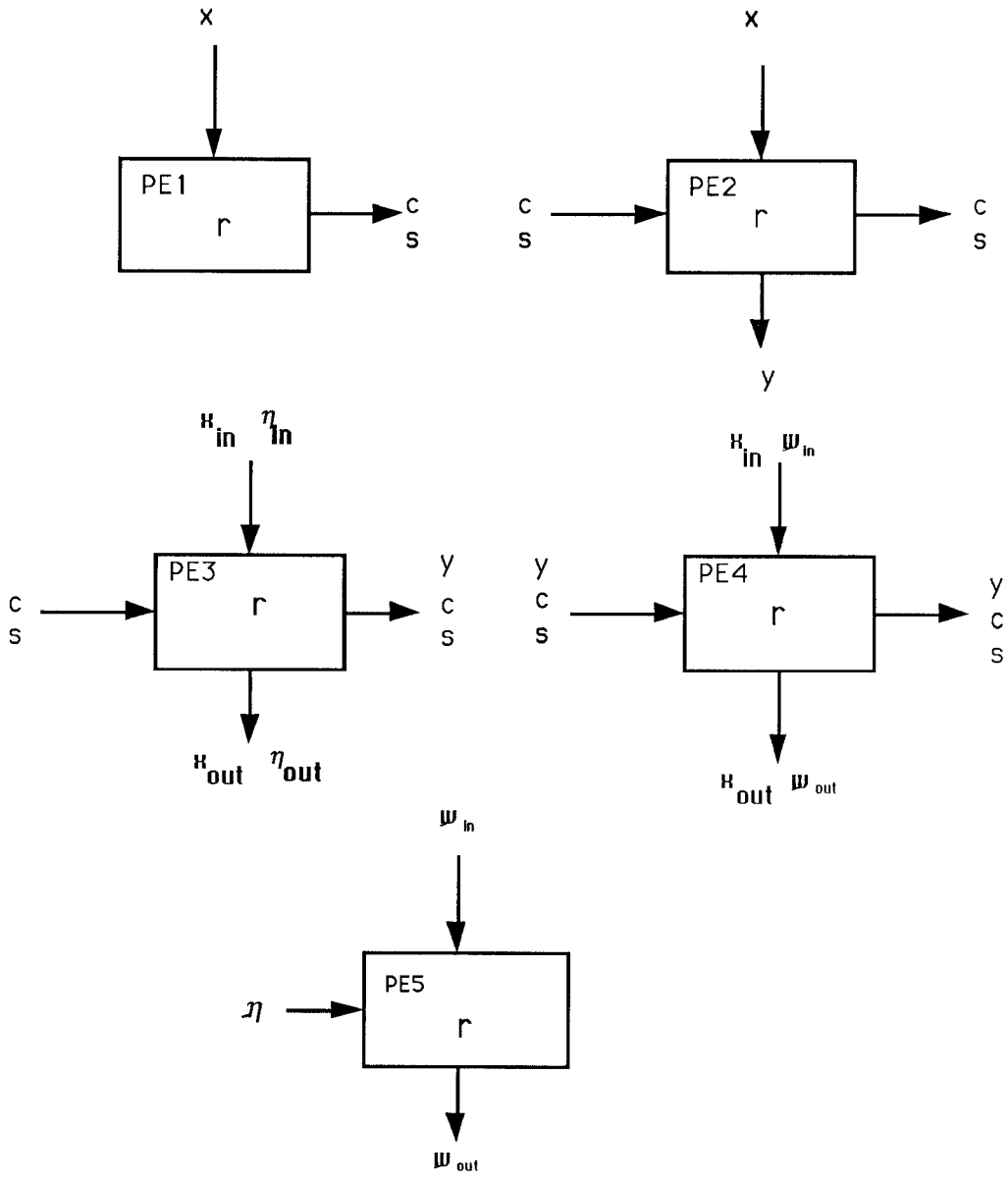


Figure 3.12: Processor Elements of CRLS Systolic Array Processor

$PE1$	$PE2$	$PE3$	$PE4$
$d \leftarrow (\beta^2 r^2 + \lvert x \rvert^2)^{\frac{1}{2}}$	$y \leftarrow -s\beta r + cx$	$x_{out} \leftarrow -\frac{1}{\beta} sr + cx_{in}$	$x_{out} \leftarrow -\frac{1}{\beta} sr + cx_{in}$
$c \leftarrow \frac{\beta r}{d}$	$r \leftarrow c\beta r + s^* x$	$r \leftarrow \frac{1}{\beta} cr + s^* x_{in}$	$r \leftarrow \frac{1}{\beta} cr + s^* x_{in}$
$s \leftarrow \frac{x}{d}$		$y \leftarrow r$	$w_{out} \leftarrow yr^* + w_{in}$
$r \leftarrow d$		$\eta_{out} \leftarrow \lvert r \rvert^2 + \eta_{in}$	
$PE5$		$w_{out} \leftarrow \frac{r w_{in}}{\beta^2 \eta}$	

Table 3.8: The Givens Based-CRLS Algorithm of Mode 1

$PE1$	$PE2$	$PE3$	$PE4$	$PE5$
$s \leftarrow \frac{x}{r}$	$y \leftarrow cx - sr$	$x_{out} \leftarrow x_{in}$	$x_{out} \leftarrow x_{in}$	$w_{out} \leftarrow w_{in}$
$c \leftarrow 1$		$if \quad x_{in} \leftarrow 1$	$if \quad x_{in} \leftarrow 1$	
		$then \quad r \leftarrow s*$	$then \quad r \leftarrow s*$	

Table 3.9: The Givens Based-CRLS Algorithm of Mode 2

To demonstrate how the parallel/pipelined weight extraction system functions, the summary of the whole system to obtain the optimal weight vector for CRLS adaptive beamforming is described in Table 3.7. The algorithm for each processor element for mode 1 of the CRLS systolic array system is provided in Table 3.8 and the algorithm for mode 2 is provided in Table 3.9. To operate the CRLS systolic array processors, the initialization procedure is used for time $0 \leq n \leq N$ to obtain the initial upper triangular matrix $R(N)$, the initial parameter vector $\underline{z}(N)$, and the lower triangular matrix $R^{-H}(N)$. Two different modes described in Table 3.8 and 3.9 are also used in the initialization procedure. The recursive procedure for parallel/pipelined weight extraction used at time $n > N$ only requires mode 1. The solution of the proposed CRLS adaptive array system only occurs in the recursive procedure. Since only one mode is required to update the weight vector recursively, the CRLS adaptive array system described should be very promising for VLSI hardware implementation and real-time high throughput array processing applications.

3.9 Fast Givens Based CRLS Algorithm

QR decomposition by the fast Givens method is easy to carry out by replacing the following matrices and vectors with a product of a diagonal matrix and

new matrices and vectors.

$$R(n) = D^{\frac{1}{2}}(n)\overline{R}(n) \tag{3.58}$$

$$\underline{z}^i(n) = D^{\frac{-1}{2}}\overline{\underline{z}}^i(n) \tag{3.59}$$

$$R^{-H}(n) = D^{\frac{-1}{2}}\overline{R}^{-H}(n) \tag{3.60}$$

In the initialization procedure, the initial upper triangular matrix can be obtained by applying a QR decomposition with the form.

$$Q(N)X(N) = R(N) \tag{3.61}$$

The initial vector $\overline{\underline{z}}^i(N)$ and the initial lower triangular matrix $\overline{R}^{-H}(N)$ are then computed as follows.

$$\left[\begin{array}{c:c} \overline{\underline{z}}^i(N) & \overline{R}^{-H}(N) \end{array}\right] = \overline{R}^{-H}(N)\left[\begin{array}{c:c} \underline{c}^i & I \end{array}\right] \tag{3.62}$$

It is required to update the system for computing the optimal weight vector. The following equation shows how the whole system can be updated to obtain the optimal weight vector.

$$\begin{aligned} \widehat{Q}(n)\left[\begin{array}{c:c:c} \beta D^{\frac{1}{2}}\overline{R}(n-1) & \frac{1}{\beta}\overline{\underline{z}}(n-1) & \frac{1}{\beta}\overline{R}^{-H}(n-1) \\ 0 & \# & \# \\ \underline{x}^T(t_n) & 0 & 0 \end{array}\right] \\ = \left[\begin{array}{c:c:c} D^{\frac{1}{2}}\overline{R}(n) & \overline{\underline{z}}^i(n) & \overline{R}^{-H}(n) \\ 0 & \# & \# \\ 0 & \# & \# \end{array}\right] \end{aligned} \tag{3.63}$$

For convenience, let

$$\underline{\widehat{w}}^i(n) = \overline{R}^{-1}(n)D^{-1}(n)\underline{\overline{z}}^i(n) \tag{3.64}$$

Finally,

$$\underline{w}^i(n) = \frac{r^i}{\underline{\overline{z}}^{iH}(n)D^{-1}(n)\underline{\overline{z}}^i(n)}\underline{\widehat{w}}^i(n) \quad for \;\; i = 1, \cdots, K \tag{3.65}$$

The algorithm and systolic array processors for the fast Givens-CRLS algorithm with modifications from those of the Givens-CRLS algorithm are described in Table 3.10 and Figures 3.13 3.14. There are five processor elements of two modes in our systolic array. The symbols of processor elements are illustrated in Figure 3.15 and the functions of each processor element in two modes are also described in Tables 3.11 and 3.12.

1. Initialize Conditions at $n = 0$ by setting

   item $D^{\frac{1}{2}}(0)\overline{R}(0) = 0 \quad \underline{\overline{z}}(0) = 0 \quad \overline{R}^{-H}(0) = 0$

2. Initialization Procedure for $0 \leq n \leq N$:

   (a) $Q(N)X(N) = D^{\frac{1}{2}}(N)\overline{R}(N)$ (Mode 1)

   (b) $\left[\underline{\overline{z}}^i(N) \ \vdots \ \overline{R}^{-H}(N) \right] = \overline{R}^{-H}(N) \left[\underline{c}^i \ \vdots \ I \right]$ (Mode 2)

3. Recursive Procedure for $n > N$ (Mode 1 only):

   (a) $\hat{Q}(n) \begin{bmatrix} \beta D^{\frac{1}{2}}(n-1)\overline{R}(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} D^{\frac{1}{2}}(n)\overline{R}(n) \\ 0 \\ 0 \end{bmatrix}$

   (b) $\hat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{\overline{z}}^i(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{\overline{z}}^i(n) \\ \# \\ \# \end{bmatrix}$

   (c) $\hat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\overline{R}^{-H}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \overline{R}^{-H}(n) \\ \# \\ \# \end{bmatrix}$

   (d) $\underline{w}(n) = \frac{r^i}{\underline{\overline{z}}^{-H}(n)D^{-1}(n)\underline{\overline{z}}}(n)\overline{R}^{-1}(n)D^{-1}(n)\underline{\overline{z}}(n)$

Table 3.10: Summary of Fast Givens-CRLS Algorithm

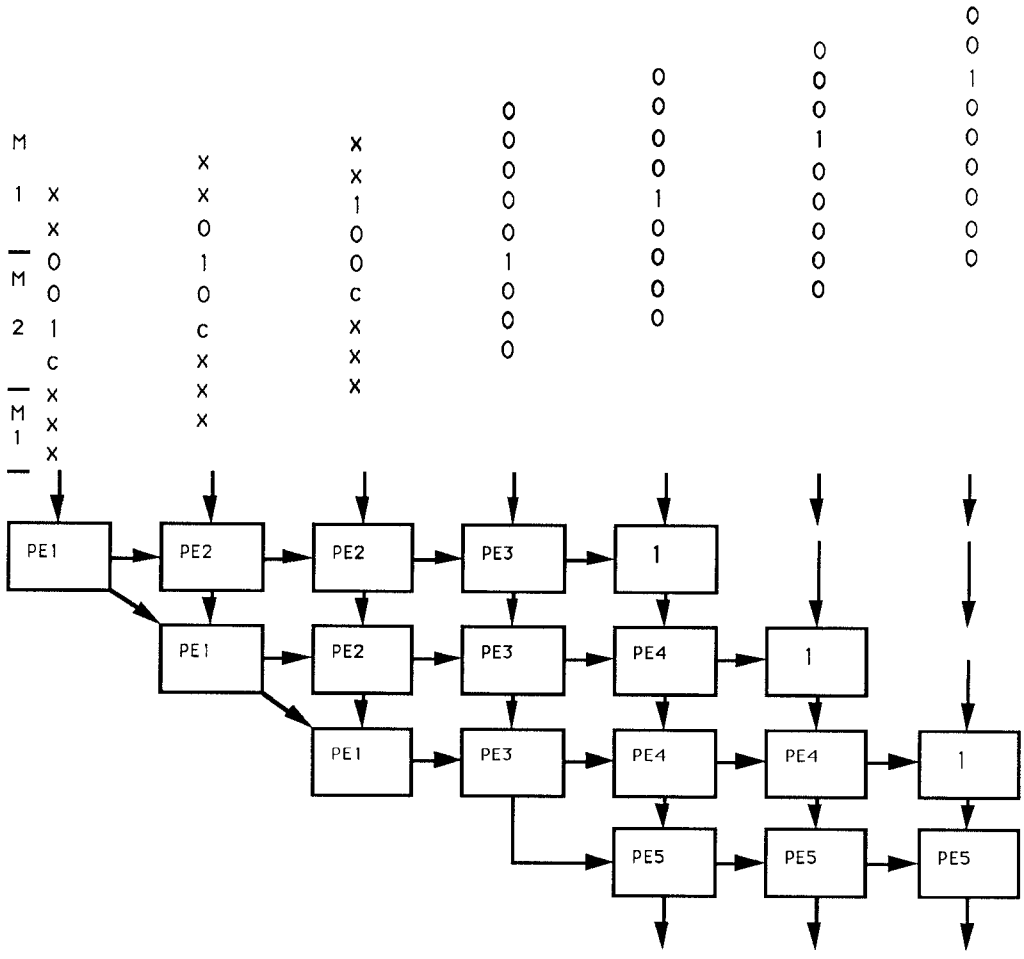


Figure 3.13: Fast CRLS Systolic Array Processors (Case 1)

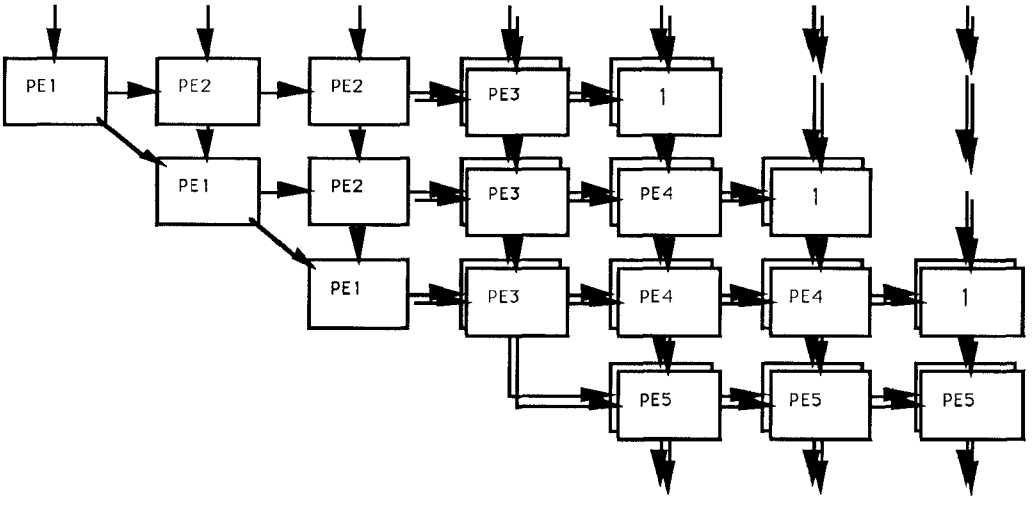


Figure 3.14: Fast CRLS Systolic Array Processors (Case 2)

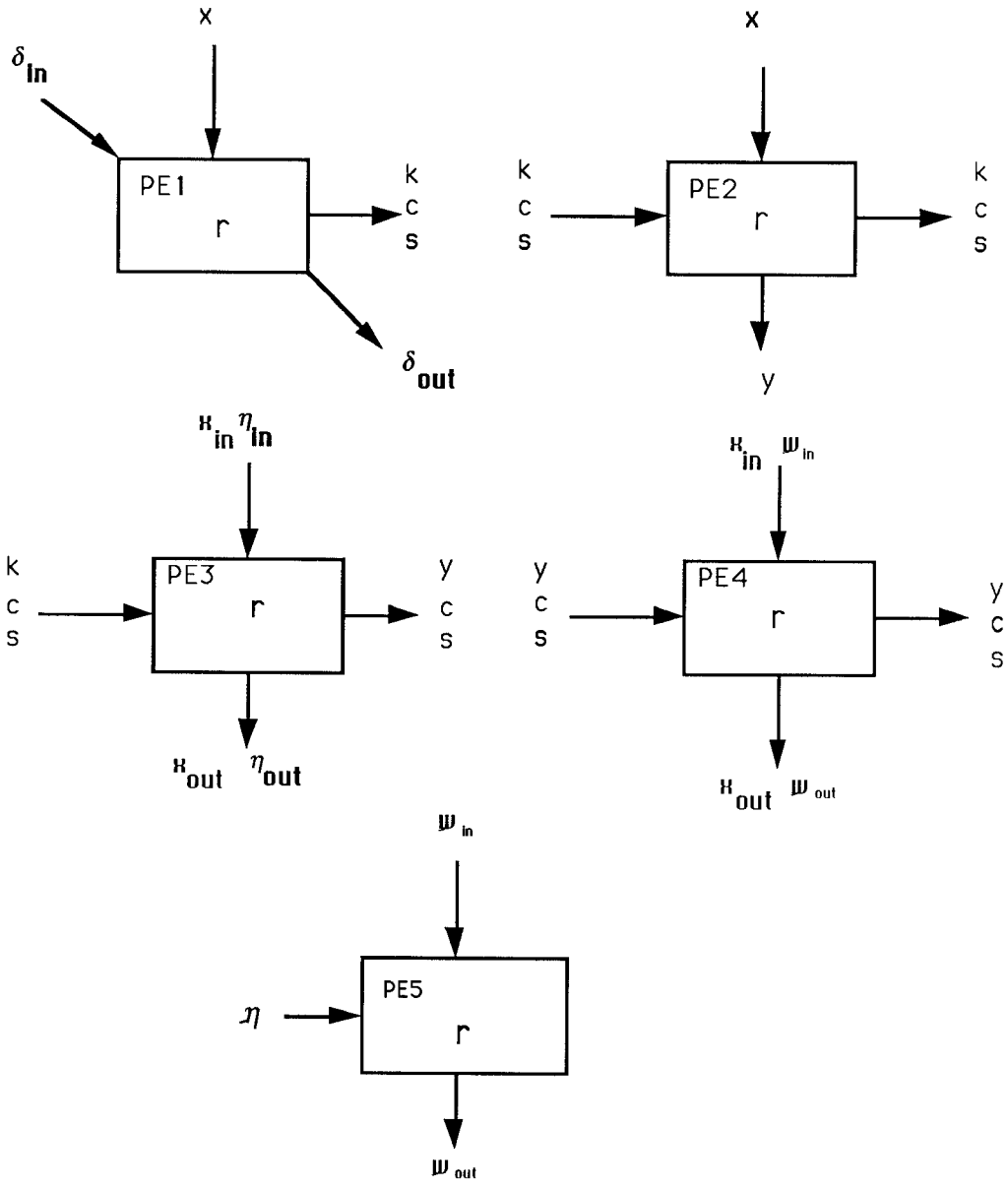


Figure 3.15: Processor Elements of CRLS Systolic Array Processor

$PE1$	$PE2$	$PE3$	$PE4$
$k \leftarrow \beta^2 r + \delta_{in}\lvert x\rvert^2$	$y \leftarrow -c\beta r + x$	$x_{out} \leftarrow -\frac{1}{\beta}cr + x_{in}$	$x_{out} \leftarrow -\frac{1}{\beta}cr + x_{in}$
$\delta_{out} \leftarrow \frac{\beta^2 r \delta_{in}}{k}$	$r \leftarrow \beta r + s^* y$	$r \leftarrow \frac{1}{\beta}r + s^* x_{out}$	$r \leftarrow \frac{1}{\beta}r + s^* x_{out}$
$c \leftarrow x$		$y \leftarrow r$	$w_{out} \leftarrow \frac{yr^*}{k} + w_{in}$
$s \leftarrow \frac{\delta_{in} x}{k}$		$\eta_{out} \leftarrow \frac{\lvert r\rvert^2}{k} + \eta_{in}$	
$r \leftarrow k$			
$PE5$		$w_{out} \leftarrow \frac{r w_{in}}{\beta^2 \eta}$	

Table 3.11: The Fast Givens-CRLS Algorithm of Mode 1

$PE1$	$PE2$	$PE3$	$PE4$	$PE5$
$s \leftarrow x$	$y \leftarrow cx - sr$	$x_{out} \leftarrow x_{in}$	$x_{out} \leftarrow x_{in}$	$w_{out} \leftarrow w_{in}$
$c \leftarrow 1$		$if \;\; x_{in} \leftarrow 1$	$if \;\; x_{in} \leftarrow 1$	
		$then \;\; r \leftarrow \frac{s*}{k}$	$then \;\; r \leftarrow s*$	

Table 3.12: The Fast Givens-CRLS Algorithm of Mode 2

Chapter 4

Fully Parallel and Pipelined CRLS Systolic Array for MVDR Beamforming

4.1 Introduction

Research in implementing the minimum variance distortionless response (MVDR) algorithm by systolic array processors to compute the residual has been very active in the last few years. This is due to the advancements in VLSI circuit technology and the demand for sophisticated systems with high throughput rate and superior numerical accuracy. The major issues in implementing the algorithm are (1) numerical stability, (2) computational efficiency, and (3)

a single fully pipelined structure. Recently, McWhirter and Shepherd [14] proposed a systolic array for a linearly constrained least-squares problem consisting of a pre-processor and canonical least-squares processor which can only handle one constraint. Yang and Bohme [15] also proposed a close-loop systolic array processors consisting of CORDIC-based processor elements with a feedback configuration for MVDR beamforming. However, those architectures are not designed as a single fully pipelined structure. Therefore, such structures may face difficulties in VLSI circuit design such as the use of available chip area and the control of propagation delays.

In a recent paper a numerically stable and computationally efficient algorithm for MVDR beamforming was introduced by Schreiber [12]. His algorithm only requires $O(N^2 + KN)$ arithmetic operations per sample time, where N is the number of sensors from an adaptive antenna array and K is the number of look direction constraints. It has robust numerical properties. However, it is difficult to implement the whole algorithm as a single fully pipelined structure as described in [28,14]. Recently, Bojanczyk and Luk [28] suggested an algorithm for MVDR beamforming which modifies McWhirter's previous work in [14], extends it to many constraints, and avoids the back substitution. Lately, McWhirter and Shepherd [14] proposed a single fully pipelined systolic array processor for MVDR beamforming by implementing Schreiber's algorithm without the need of an extra back substitution processor for computing the

residual. Moreover, they also presented an algorithm for MVDR beamforming using the recursive least squares algorithm proposed in McWhirter's previous paper [24]. According to McWhirter and Shepherd's paper, by comparing the residual for MVDR and RLS algorithms, it seems easy to obtain MVDR beamforming from the recursive least squares algorithm by forcing the desired input data to be zero. Unfortunately, the residual of the recursive least squares algorithm described in [24] is derived by employing the lower part of a unitary matrix, S, while the residual of MVDR beamforming is obtained by using the upper part of a unitary matrix, P, where the upper part and lower part of the unitary matrix $Q(n)$ is defined as $Q(n) = \begin{bmatrix} P(n) \\ S(n) \end{bmatrix}$. Therefore, the algorithm for MVDR beamforming can not be obtained directly by adopting the recursive least squares algorithm.

In this chapter, the problem of MVDR beamforming using an RLS systolic array is addressed and a new algorithm for MVDR beamforming to compute the residual is also presented. In the next section Mcwhirter's RLS and MVDR algorithms are summarized and the problem of using the RLS algorithm for MVDR beamforming to compute the residual is discussed. In the third section a newly developed MVDR beamforming algorithm with single fully pipelined systolic array processors is described in detail. Mcwhirter's MVDR and our newly developed MVDR beamforming methods are compared. Compared to McWhirter's MVDR method, the new MVDR beamforming algorithm not only

has the advantage with a single fully pipelined structure but also is numerically stable.

4.2 Comparison Between McWhirter's RLS and MVDR Algorithms

In this section, the problem of using the RLS algorithm for MVDR beamforming is addressed and McWhirter's RLS algorithm and MVDR adaptive beamforming are summarized. We claim that MVDR beamforming can not use RLS algorithm simply by driving the desired data $z(t_n)$ to zero. The correct algorithm for McWhirter's MVDR beamforming will be rederived since the RLS algorithm can not be used directly. To understand why the solution of the MVDR algorithm for computing the residual can not use the RLS algorithm, McWhirter's famous RLS systolic arrays and MVDR systolic arrays are necessarily examined in detail. Comparison between the two McWhirter architectures is very important to get a clear picture of the approach described in this chapter. Therefore, we start a the detailed description of McWhirter's work and then present a better solution for computing the residual in MVDR beamforming. In the next two subsections Mcwhirter's algorithm for RLS approximation is summarized and the MVDR algorithm for computing the residual by driving the desired data to be zero is described [24].

4.2.1 McWhirter's RLS Algorithm

McWhirter's RLS algorithm is summarized as follows. The $n \times 1$ residual vector for the least squares problem is

$$\underline{e}(n) = X(n)\underline{w}(n) - \underline{z}(n) \tag{4.1}$$

where $X(n)$, is a given $n \times p$ observed data matrix

$$X^T(n) = B(n) \left[\begin{array}{cccc} \underline{x}(t_1) & \underline{x}(t_2) & \cdots & \underline{x}(t_n) \end{array} \right]$$

$\underline{z}(n)$, is a given $n \times 1$ desired data vector

$$\underline{z}^T(n) = B(n) \left[\begin{array}{cccc} z(t_1) & z(t_2) & \cdots & z(t_n) \end{array} \right]$$

$\underline{w}(n)$, is a given $p \times 1$ weight vector

$$\underline{w}^T(n) = \left[\begin{array}{cccc} w_1(t_n) & w_2(t_n) & \cdots & w_N(t_n) \end{array} \right]$$

$\underline{e}(n)$, the residual vector, is

$$\underline{e}(n)^T = \left[\begin{array}{cccc} e(t_1) & e(t_2) & \cdots & e(t_n) \end{array} \right]$$

and where $B(n)$, an exponential forget factor which emphasizes the statistical weight on the new data and gradually scales down the old data in the least squares computation, is given by

$$B(n) = \left[\begin{array}{ccccc} \beta^{n-1} & 0 & 0 & 0 & 0 \\ 0 & \beta^{n-2} & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

A QR decomposition can be applied to the data matrix $X(n)$, so that

$$Q(n)X(n) = \begin{bmatrix} R(n) \\ 0 \end{bmatrix} \tag{4.2}$$

The unitary matrix $Q(n)$ can be partitioned into two submatrices which are a $p \times n$ matrix $P(n)$ and a $(n-p) \times n$ matrix $S(n)$ as follows

$$Q(n) = \begin{bmatrix} P(n) \\ S(n) \end{bmatrix} \tag{4.3}$$

The QR decomposition can be also be applied to the desired data vector $\underline{z}$. Then

$$\begin{aligned} Q(n)\underline{z}(n) &= \begin{bmatrix} P(n)\underline{z}(n) \\ S(n)\underline{z}(n) \end{bmatrix} \\ &= \begin{bmatrix} \underline{u}(n) \\ \underline{v}(n) \end{bmatrix} \end{aligned} \tag{4.4}$$

The L_2 norm of the residual is

$$\begin{aligned} ||\underline{e}(n)|| &= ||X(n)\underline{w}(n) - \underline{z}(n)|| \\ &= ||Q^H(\begin{bmatrix} R(n) \\ 0 \end{bmatrix}\underline{w}(n) - \begin{bmatrix} \underline{u}(n) \\ \underline{v} \end{bmatrix})|| \\ &= ||\begin{bmatrix} R(n)\underline{w}(n) - \underline{u}(n) \\ -\underline{v} \end{bmatrix}|| \end{aligned} \tag{4.5}$$

Therefore, the least-squares weight vector $\underline{w}_{LS}(n)$ must satisfy the equation

$$\underline{w}_{LS}(n) = R^{-1}(n)\underline{u}(n) \tag{4.6}$$

Hence the minimized $||\underline{e}||_2$ is

$$||\underline{e}(n)||_2 = ||\underline{v}(n)||_2 \tag{4.7}$$

where $\underline{v}(n) = S(n)\underline{z}(n)$.

It was shown in [24] that the unitary matrix $Q(n)$ which can be applied to update the whole system is

$$Q(n) = \hat{Q}(n)\overline{Q}(n-1) \tag{4.8}$$

where $\overline{Q}(n-1)$ is

$$\overline{Q}(n-1) = \begin{bmatrix} P(n-1) & 0 \\ S(n-1) & 0 \\ 0 & 1 \end{bmatrix} \tag{4.9}$$

and $\hat{Q}(n)$ is

$$\hat{Q}(n) = \begin{bmatrix} A(n) & 0 & \underline{a} \\ 0 & I & 0 \\ \underline{b}^T & 0 & \gamma \end{bmatrix} \tag{4.10}$$

Therefore, the submatrices $P(n)$ and $S(n)$ of unitary matrix $Q(n)$ are

$$P(n) = \begin{bmatrix} A(n)P(n-1) & \underline{a}(n) \end{bmatrix} \tag{4.11}$$

and

$$S(n) = \begin{bmatrix} S(n-1) & 0 \\ \underline{b}^T(n)P(n-1) & \gamma \end{bmatrix} \tag{4.12}$$

By substituting Equation 4.6 into Equation 4.1, the residual vector is found to be

$$\underline{e}_{LS}(n) = X(n)R^{-1}(n)\underline{u}(n) - \underline{z}(n) \tag{4.13}$$

The n^{th} element of the residual vector of recursive least squares described by McWhirter is given as follows.

$$\underline{e}_{LS}(n) = Q^H(n)\begin{bmatrix} R(n) \\ 0 \end{bmatrix} R^{-1}(n)\underline{u}(n) - Q^H(n)\begin{bmatrix} \underline{u}(n) \\ \underline{v}(n) \end{bmatrix} \tag{4.14}$$

Therefore, the residual vector has the form

$$\begin{aligned} \underline{e}_{LS}(n) &= Q^H(n)\begin{bmatrix} 0 \\ \underline{v}(n) \end{bmatrix} \\ &= S^T(n)\underline{v}(n) \end{aligned} \tag{4.15}$$

where $S^T(n)$ has the form

$$S^T(n) = \begin{bmatrix} S^T(n-1) & P^T(n-1)\underline{b}(n) \\ 0 & \gamma \end{bmatrix},$$

and $\underline{v}(n)$ has the form

$$\underline{v}(n) = \begin{bmatrix} \beta\underline{v}(n-1) \\ \alpha(n) \end{bmatrix}.$$

The n^{th} element of the residual vector $\underline{e}_{LS}$ described in [24] has the form

$$e(t_n) = \gamma(n)\alpha(n) \tag{4.16}$$

Equation 4.15 shows that the residual for the recursive least squares problem is obtained from the lower part of the unitary matrix $S(n)$. Notice that

the upper part of unitary matrix, $P(n)$, is not used to compute the residual. We will describe in the next section that without the desired data the upper part of the unitary matrix $P(n)$ is used to compute the residual for the case of MVDR beamforming. According to the RLS algorithm discussed in this section, McWhirter's systolic array for the recursive least-squares problem illustrated in [24] is shown in Figure 4.1.

4.2.2 McWhirter's MVDR Beamforming

MVDR beamforming can be formulated as follows. The $n \times 1$ residual vector for the MVDR problem is

$$\underline{e}(n) = X(n)\underline{w}(n) \tag{4.17}$$

where $X(n)$ and $\underline{w}(n)$ are the same as in the least squares problem.

The MVDR problem is to find $\underline{w}(n)$ so that

$$\min_{\underline{w}(n)} \underline{w}^H(n) M \underline{w}(n)$$

$$\text{subject to } \underline{c}^{iH}\underline{w}(n) = r^i \quad for \;\; i = 1, 2, \cdots, K.$$

Using the method of Lagrange multipliers the optimal weight vector $\underline{w}^i_{opt}$ is

$$\underline{w}^i_{opt} = \frac{r^i M^{-1}\underline{c}^i}{\underline{c}^{iH} M^{-1}\underline{c}^i} \quad for \;\; i = 1, \cdots, K. \tag{4.18}$$

where

$$M = X^H X \tag{4.19}$$

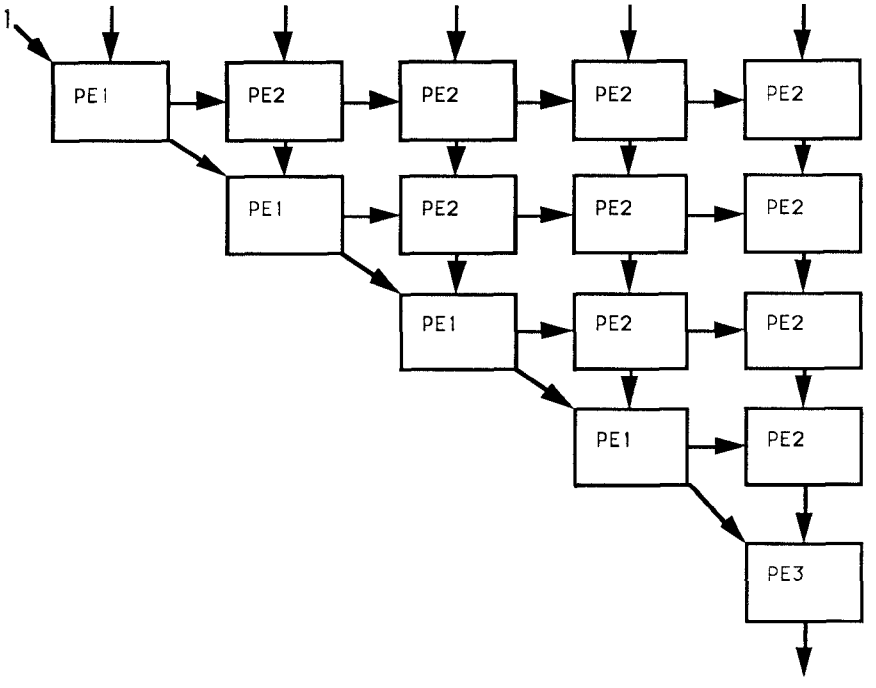


Figure 4.1: McWhirter's RLS Systolic Array Processor [11]

By applying QR decomposition to the input observed data $X(n)$, we have

$$Q(n)X(n) = \begin{bmatrix} R(n) \\ \\ 0 \end{bmatrix} \tag{4.20}$$

Substituting Equation 4.20 into 4.19, we have

$$M(n) = R^H(n)R(n) \tag{4.21}$$

Therefore, the optimal weight vector is

$$\underline{w}^i_{opt} = \frac{r^i R^{-1}(n) R^{-H}(n) \underline{c}^i}{\underline{c}^{iH} R^{-1}(n) R^{-H}(n) \underline{c}^i} \quad for \;\; i = 1, \cdots, K. \tag{4.22}$$

Let us define a parameter vector $\underline{z}(n)$ given by

$$\underline{z}(n) = R^{-H}(n)\underline{c}^i \tag{4.23}$$

On substituting Equation 4.23 into 4.22, the residual vector for MVDR beamforming is

$$\underline{e}(n) = \frac{r^i}{||\underline{z}(n)||^2} X(n) R^{-1}(n) \underline{z}(n) \tag{4.24}$$

For convenience, define a vector $\underline{\hat{e}}(n)$ as

$$\underline{\hat{e}}(n) = X(n)R^{-1}(n)\underline{z}(n) \tag{4.25}$$

Unfortunately, rather than directly computing Equation 4.25 for the n^{th} term of the residual vector, the following equation is used to match the RLS algorithm by driving the desired data to zero in McWhirter and Shepherd's paper [14].

$$\hat{e}_{MVDR}(t_n) = \underline{x}^T(t_n) R^{-1}(n) \underline{z}(n) \tag{4.26}$$

By changing the data matrix $X(n)$ into a data vector at the n^{th} snapshot in Equation 4.13, the n^{th} residual of the RLS algorithm can be written as

$$e_{RLS}(t_n) = \underline{x}^T(t_n)R^{-1}(n)\underline{u}(n) - z(t_n) \tag{4.27}$$

Comparing to Equations 4.26 and 4.27, it seems that by forcing the desired data $z(t_n)$ to be zero, the residual for MVDR beamforming can be obtained by using the RLS algorithm. Therefore, according to Equation 4.16, the residual for MVDR beamforming seems to be

$$\begin{aligned} \widehat{e}_{MVDR}(t_n) &= \gamma(n)\alpha(n) \\ &= c_n c_{n-1} \cdots c_2 c_1 \alpha(n) \end{aligned} \tag{4.28}$$

Comparing the two residual equations for RLS and MVDR shown in [14], McWhirter's MVDR beamforming seems to have the same residual for RLS by driving the desired data to zero. Therefore, it looks as if the MVDR systolic array could employ the RLS systolic array by inputting zeroes. In [14], the systolic array processor for MVDR beamforming has the structure given by Figure 4.2.

The question arises "Can MVDR beamforming use the RLS algorithm by driving the desired data to zero to compute the residual?" The question will be answered negatively because given the zero desired data, the algorithm to compute the n^{th} residual is not the same as McWhirter's RLS algorithm described in this section. By a simple derivation in the next section, we point out

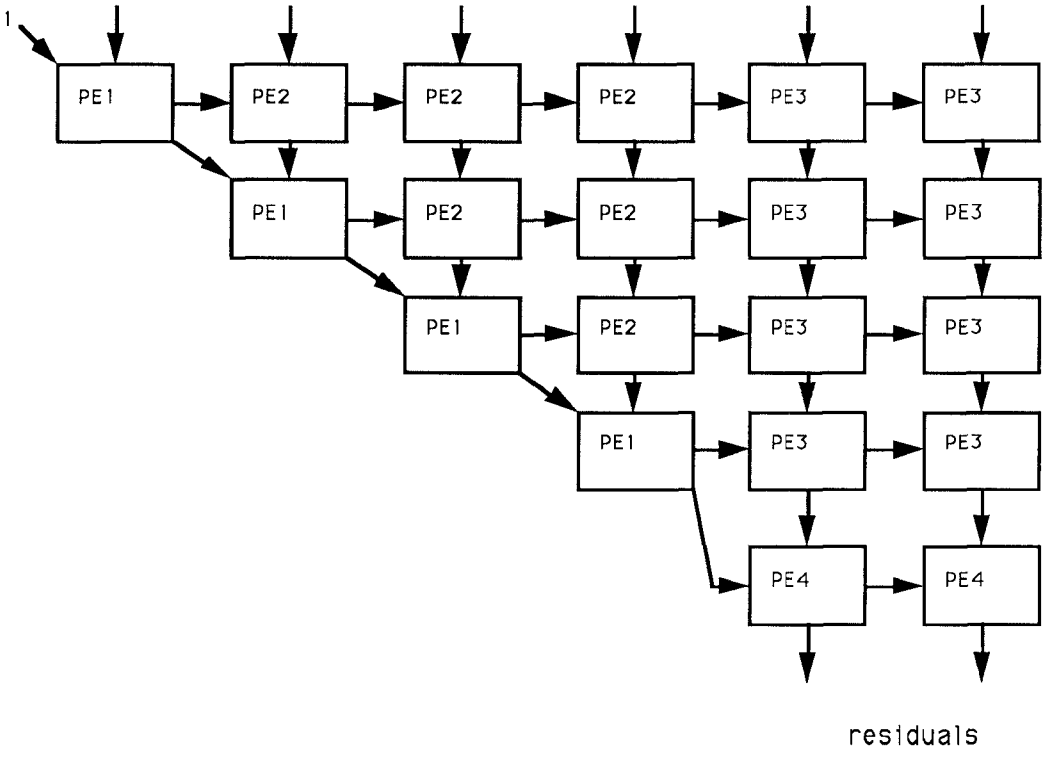


Figure 4.2: McWhirter's MVDR Systolic Array Processors [14]

that the algorithm for McWhirter's MVDR beamforming, using Equation 4.28 to compute the residual by forcing the desired data $z(t_n)$ in the Equation 4.27 to be zero, is not correct. The modified algorithm to compute the n^{th} element of the residual vector for MVDR beamforming will be given in the next section.

4.3 A Novel MVDR algorithm

Recall that the residual vector for MVDR beamforming is

$$\underline{e}(n) = \frac{r^i}{||\underline{z}(n)||^2} X(n) R^{-1}(n) \underline{z}(n) \tag{4.29}$$

For convenience, define a vector $\hat{\underline{e}}(n)$ as

$$\hat{\underline{e}}(n) = X(n) R^{-1}(n) \underline{z}(n) \tag{4.30}$$

By substituting Equation 4.2 into 4.30, we have

$$\begin{aligned} \hat{\underline{e}}(n) &= Q^H(n) \begin{bmatrix} R(n) \\ 0 \end{bmatrix} R^{-1}(n) \underline{u}(n) \\ &= Q^H(n) \begin{bmatrix} \underline{z}(n) \\ 0 \end{bmatrix} \\ &= P^H(n) \underline{z}(n) \end{aligned} \tag{4.31}$$

where $P^H(n)$ is

$$P^H(n) = \begin{bmatrix} P^H(n-1) A^H(n) \\ \underline{a}^H(n) \end{bmatrix}.$$

Therefore, the n^{th} element of the residual vector for the case of MVDR beamforming has the form

$$\widehat{e}(t_n) = \underline{a}^H(n)\underline{z}(n) \tag{4.32}$$

where $\underline{a}^H(n)$ has the form

$$\underline{a}^H(n) = \begin{bmatrix} s_1 & s_2c_1 & s_3c_2c_1 & \cdots & s_Nc_{N-1}\cdots c_1 \end{bmatrix}.$$

Comparing Equations 4.15 and 4.31, we see that instead of using McWhirter's RLS algorithm, the MVDR beamforming should use our newly developed algorithm for computing the residual. The n^{th} element of the residual vector is

$$e(t_n) = \frac{r^i}{||\underline{z}(n)||^2}\underline{a}^H(n)\underline{z}(n) \tag{4.33}$$

The solution obtained by the MVDR algorithm for the residual is used only for time $n \geq N$ for which the data matrix $X(n)$ is of full rank. However, the initial constant which sets the whole system to be zero requires an initialization algorithm. The exact initialization procedure is used for the period $0 \leq n \leq N$. Initialization algorithm must be used first before the recursive algorithm is employed. The initialization procedure and the recursive procedure are described in the following subsection.

4.3.1 Initialization Procedure

The initialization procedure for data samples taken for $0 \leq n \leq N$ will now be introduced to obtain the initial upper triangular matrix $R(N)$ and a vector $\underline{z}(N)$. Mode 1 and mode 2 for computing the upper triangular matrix $R(N)$ and a parameter vector $\underline{z}(N)$ are described.

The QR decomposition applied to the first N data snapshots to obtain the initial upper triangular matrix under mode 1 has the form

$$Q(N)X(N) = R(N) \tag{4.34}$$

The forward substitution under mode 2 has the form

$$\underline{z}(N) = R^{-H}(N)\underline{c}^{i} \tag{4.35}$$

4.3.2 Recursive Updating

The recursive algorithm is applied for time $n > N$ to update and compute the residual. It is well known [14] that the QR decomposition of the observed data $X(n)$ can be implemented recursively on a triangular systolic array. The upper triangular matrix can be updated by using the QR decomposition given by

$$\widehat{Q}(n)\begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^{T}(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix} \tag{4.36}$$

where $\widehat{Q}(n)$ has the form

$$\widehat{Q}(n) = \begin{bmatrix} A(n) & 0 & \underline{a}(n) \\ 0 & I & 0 \\ \underline{b}^T(n) & 0 & \gamma(n) \end{bmatrix}.$$

The matrix $A(n)$, the vector $\underline{a}(n)$, the vector $\underline{b}^T$, and a scalar $\gamma(n)$ have the following forms

$$A(n) = \begin{bmatrix} c_1 & 0 & 0 & \cdots & 0 \\ -s_2^* & c_2 & 0 & \cdots & 0 \\ -s_4^* c_3 c_2 s_1 & -s_4^* c_3 s_2 & -s_4^* s_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -s_n^* c_{n-1} \cdots c_2 s_1 & -s_n^* c_{n-1} \cdots c_3 s_2 & -s_n^* c_{n-1} \cdots c_4 s_3 & \cdots & c_n \end{bmatrix}, \tag{4.37}$$

$$\underline{a}^T(n) = \begin{bmatrix} s_1^* & s_2^* c_1 & s_3^* c_2 c_1 & \cdots & s_n^* c_{n-1} \cdots c_1 \end{bmatrix}, \tag{4.38}$$

$$\underline{b}^T(n) = \begin{bmatrix} -c_n c_{n-1} \cdots c_2 s_1 & -c_n \cdots c_3 s_2 & -c_n \cdots c_4 s_3 & \cdots & -s_n \end{bmatrix}, \tag{4.39}$$

and,

$$\gamma(n) = c_n c_{n-1} \cdots c_2 c_1 \tag{4.40}$$

Since

$$\begin{aligned} \underline{c}^i &= R^H(n)\underline{z}(n) \\ &= R^H(n-1)\underline{z}(n-1) \end{aligned}$$

$$= \begin{bmatrix} \beta R^H(n-1) & 0 & \underline{x}^*(t_n) \end{bmatrix} \widehat{Q}^H(n)\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{z}(n-1) \\ \# \\ 0 \end{bmatrix} \quad (4.41)$$

The same QR decomposition can be applied to update the parameter $\underline{z}(n-1)$.

$$\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{z}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{z}(n) \\ \# \\ \# \end{bmatrix} \quad (4.42)$$

where # denotes an arbitrary vector of no interest in mathematical and physical concept.

Recall from Equation 4.33 that the n^{th} element of the residual vector is

$$\begin{aligned} e(t_n) &= \frac{r^i}{||\underline{z}(n)||^2}\underline{a}^H(n)\underline{z}(n) \\ &= \frac{r^i}{||\underline{z}(n)||^2}\sum_{j=1}^{n} s_j c_{j-1}\cdots c_{j-(j-1)} z_j \end{aligned} \quad (4.43)$$

In this chapter, we found that the equation for the n^{th} element of the residual proposed by McWhirter and Shepherd is incorrect. According to the discussion section in [24], McWhirter's algorithm requires reinitialization of the MVDR system every 5000 samples to retain sufficient accuracy for most practical applications. In conclusion, McWhirter's algorithm for MVDR beamforming using the RLS algorithm has an error in computing the residual which creates the numerical instability. A robust numerically stable and computationally efficient MVDR algorithm which is mapped into a single fully pipelined

systolic array is described in this section. In the next section, a state of the art systolic array processor is designed for MVDR beamforming.

4.4 MVDR Systolic Array Processors

In this section, first, the newly developed MVDR algorithm is summarized in Table 4.1. Then, a single fully pipelined structure is described for the case of four sensors and two constraints in Figure 4.3. To operate the MVDR systolic array processors, the initialization procedure is required for time $0 \leq n \leq N$ to obtain the initial upper triangular $R(N)$ and the initial parameter vector $\underline{z}(N)$. Two different modes shown in Tables 4.2 and 4.3 are introduced to form the initial upper triangular matrix and initial parameter vector. The recursive procedure used for $n > N$ only requires mode 1 to obtain the residuals. In Figure 4.4, the symbols for four processor elements are depicted and in Tables 4.2 and 4.3, the algorithms for mode 1 and mode 2 are described in detail.

4.5 Fast Givens Based-MVDR Beamforming

The QR decomposition by the fast Givens method is easy to carry out by replacing the following matrices and vectors with a product of diagonal matrix and new matrices and vectors as shown in the next equations

$$R(n) = D^{\frac{1}{2}}(n)\overline{R}(n) \tag{4.44}$$

1. Initialize Conditions at $n = 0$ by setting

   $R(0) = 0 \quad \underline{z}(0) = 0 \quad R^{-H}(0) = 0$

2. Initialization Procedure for $0 \leq n \leq N$:

   (a) $Q(N)X(N) = R(N)$ (Mode 1)

   (b) $\underline{z}^i(N) = R^{-H}(N)\underline{c}^i$ (Mode 2)

3. Recursive Procedure for $n > N$ (Mode 1 only):

   (a) $$\widehat{Q}(n) \begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix}$$

   (b) $$\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{z}^i(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{z}^i(n) \\ \# \\ \# \end{bmatrix}$$

   (c) $e(t_n) = \frac{r^i}{||\underline{z}(n)||^2} \sum_{j=1}^{n} s_j c_{j-1} \cdots c_{j-(j-1)} z_j$

Table 4.1: Summary of Parallel/Pipelined QRD-MVDR Algorithm

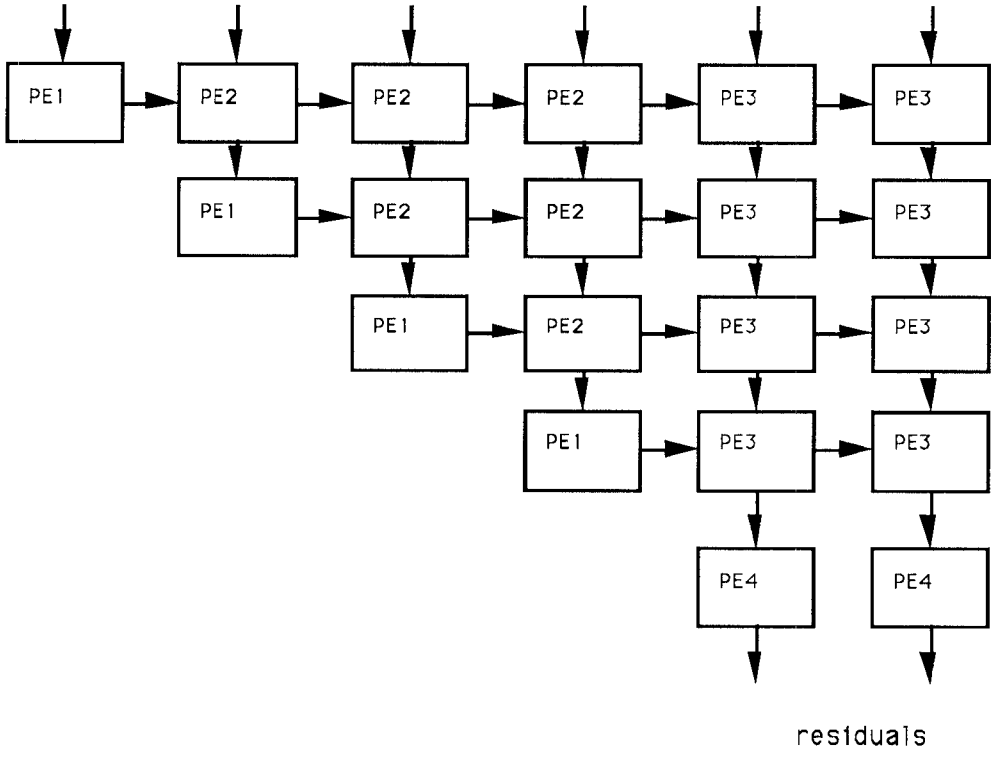


Figure 4.3: MVDR Systolic Array Processor

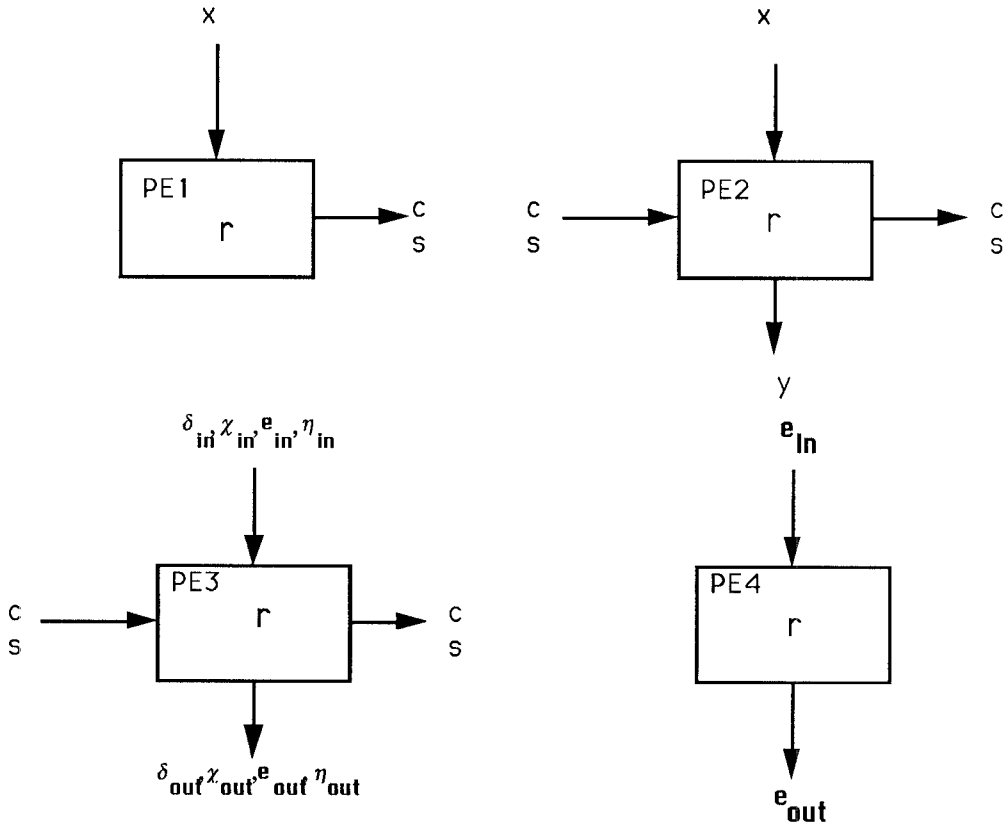


Figure 4.4: Processor Elements of MVDR Systolic Array

$PE1$	$PE2$	$PE3$	$PE4$
$d \leftarrow (\beta^2 r^2 + \lvert x \rvert^2)^{\frac{1}{2}}$	$y \leftarrow -s\beta r + cx$	$x_{out} \leftarrow -s\frac{1}{\beta} r + cx_{in}$	$e \leftarrow \frac{re_{in}}{\beta^2 \eta}$
$c \leftarrow \frac{\beta r}{d}$	$r \leftarrow c\beta r + s^* x$	$r \leftarrow c\frac{1}{\beta} r + s^* x_{in}$	
$s \leftarrow \frac{x}{d}$		$\eta_{out} \leftarrow \lvert r \rvert^2 + \eta_{in}$	
$r \leftarrow d$		$\delta_{out} \leftarrow c\delta_{in}$	
		$e_{out} \leftarrow s\delta_{in} r + e_{in}$	

Table 4.2: The Algorithm for Mode 1

$PE1$	$PE2$	$PE3$	$PE4$
$s \leftarrow \frac{x}{r}$	$y \leftarrow cx - sr$	$x_{out} \leftarrow x_{in}$	$e_{out} \leftarrow e_{in}$
$c \leftarrow 1$		$if \quad x_{in} \leftarrow 1$	
		$then \quad r \leftarrow s*$	

Table 4.3: The Algorithm for Mode 2

$$\underline{z}^i(n) = D^{\frac{-1}{2}}\underline{\overline{z}}^i(n) \tag{4.45}$$

For the initialization procedure, the initial upper triangular is first computed as

$$Q(N)X(N) = D^{\frac{1}{2}}(N)R(N) \tag{4.46}$$

and the initial vector $\underline{\overline{z}}^i(N)$ is then computed as

$$\underline{\overline{z}}^i(N) = \overline{R}^{-H}(N)\underline{c}^i \tag{4.47}$$

Update the system is required for computing the residual. The following equation shows how to update the whole system together.

$$\begin{aligned}\widehat{Q}(n)\begin{bmatrix} \beta D^{\frac{1}{2}}(n-1)\overline{R}(n-1) & \vdots & \frac{1}{\beta}\underline{\overline{z}}(n-1) \\ 0 & \vdots & \# \\ \underline{x}^T(t_n) & \vdots & 0 \end{bmatrix} \\ = \begin{bmatrix} D^{\frac{1}{2}}(n)\overline{R}(n) & \vdots & \underline{\overline{z}}^i(n) \\ 0 & \vdots & \# \\ 0 & \vdots & 0 \end{bmatrix}\end{aligned} \tag{4.48}$$

According to Equation 4.43, for convenience, let us define $\widehat{\underline{e}}^i(n)$ as

$$\widehat{\underline{e}}^i(n) = \underline{a}^H(n)D^{-1}(n)\underline{\overline{z}}^i(n) \tag{4.49}$$

Finally

$$\underline{e}^i(n) = \frac{r^i}{\underline{\overline{z}}^{iH}(n)D^{-1}(n)\underline{\overline{z}}^i(n)}\widehat{\underline{e}}^i(n) \quad for \;\; i = 1, \cdots, K \tag{4.50}$$

The algorithm and systolic array processors for the fast Givens-RCLS with modifications from those of the Givens-RCLS are described in Tables 4.4 and Figures 4.5. The four processor elements of the fast Givens MVDR systolic array are depicted in Figure 4.6 and their algorithms for mode 1 and mode 2 are stated in Table 4.5 and 4.6.

1. Initialize Conditions at $n = 0$ by setting

   $D^{\frac{1}{2}}(0)R(0) = 0 \quad \underline{z}(0) = 0 \quad R^{-H}(0) = 0$

2. Initialization Procedure for $0 \leq n \leq N$:

   (a) $Q(N)X(N) = D^{\frac{1}{2}}(N)R(N)$ (Mode 1)

   (b) $\underline{z}^i(N) = R^{-H}(N)\underline{c}^i$ (Mode 2)

3. Recursive Procedure for $n > N$ (Mode 1 only):

   (a) $$\widehat{Q}(n)\begin{bmatrix} \beta D^{\frac{1}{2}}(n-1)R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix}$$

   (b) $$\widehat{Q}(n)\begin{bmatrix} \frac{1}{\beta}\underline{z}^i(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{z}^i(n) \\ \# \\ \# \end{bmatrix}$$

   (c) $e(t_n) = \frac{r^i}{\underline{z}^H(n)D^{-1}(n)\underline{z}(n)} \sum_{j=1}^{n} s_j c_{j-1} \cdots c_{j-(j-1)} \frac{z_j}{d(j)}$

Table 4.4: Summary of Parallel/Pipelined Fast Givens-MVDR Algorithm

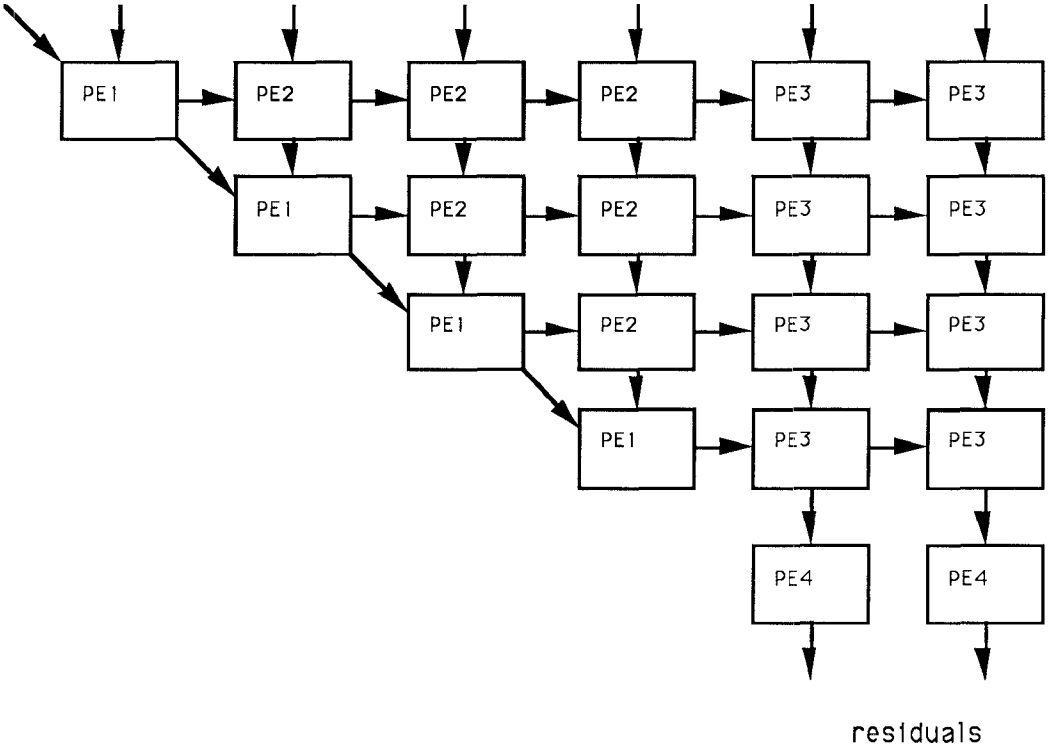


Figure 4.5: Fast MVDR Systolic Array Processor

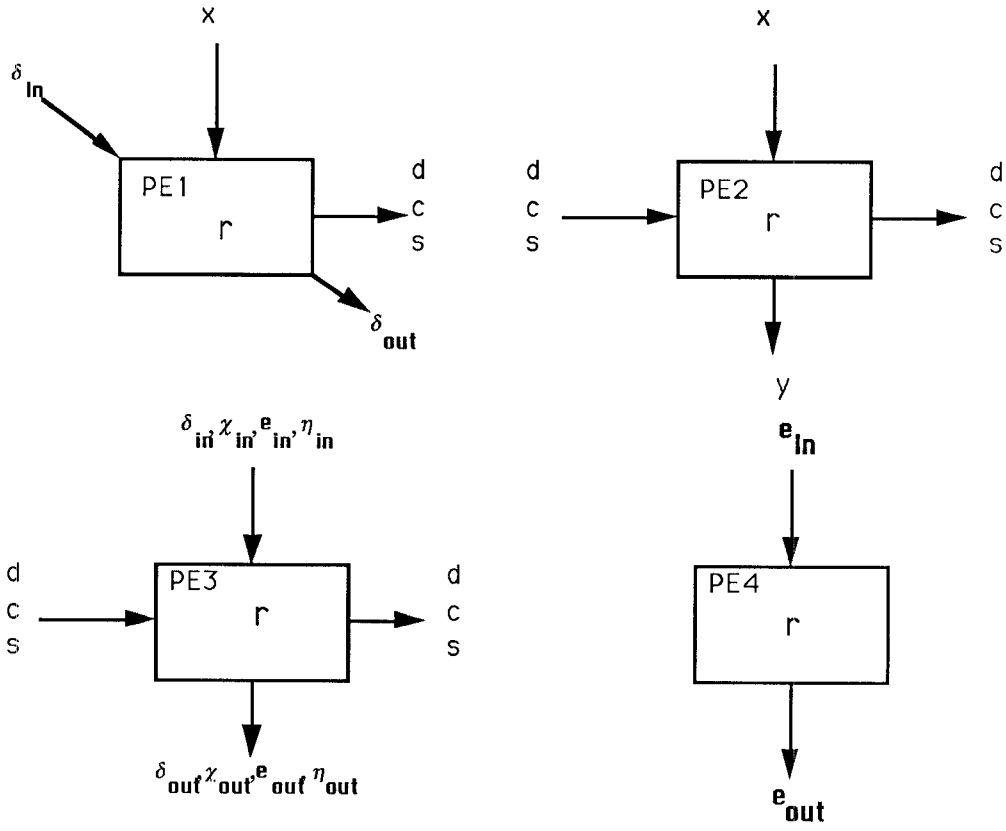


Figure 4.6: Processor Element of Fast MVDR Systolic Array

$PE1$	$PE2$	$PE3$	$PE4$
$d \leftarrow \beta^2 r^2 + \delta \lvert x \rvert^2$	$y \leftarrow -c\beta r + x$	$x_{out} \leftarrow -c\frac{1}{\beta} r + x_{in}$	$e \leftarrow \frac{r\alpha^2 e_{in}}{\eta}$
$c \leftarrow \frac{\beta r}{d}$	$r \leftarrow \beta r + s^* y$	$r \leftarrow \frac{1}{\beta} r + s^* x_{out}$	
$s \leftarrow \frac{x}{d}$		$\eta_{out} \leftarrow \frac{\lvert r \rvert^2}{d} + \eta_{in}$	
$r \leftarrow d$		$\delta_{out} \leftarrow c\delta_{in}$	
		$e_{out} \leftarrow s\delta_{in}\frac{r}{d} + e_{in}$	

Table 4.5: The Fast Algorithm of Mode 1

$PE1$	$PE2$	$PE3$	$PE4$
$s \leftarrow \frac{x}{r}$	$y \leftarrow cx - sr$	$x_{out} \leftarrow x_{in}$	$e_{out} \leftarrow e_{in}$
$c \leftarrow 1$		$if x_{in} \leftarrow 1$	
		$then r \leftarrow \frac{s*}{d}$	

Table 4.6: The Fast Algorithm of Mode 2

Chapter 5

VLSI Algorithms and Architectures for Complex Householder Transformation with Application to Array Processing

5.1 Introduction

Mapping QR decomposition algorithms onto systolic arrays has received considerable attention recently. There are several reasons. One of them is that

recent developments in VLSI technology make it possible to build a multiprocessor system on a chip. Another reason is that many real-time signal processing applications require both a high throughput rate and superior numerical accuracy. Therefore, the combination of numerical analysis problems and VLSI designs has played an important role in real-time applications of modern signal processing due to the increasing use of numerical analysis in these areas. Recently, many issues such as systolic Cholesky decomposition [63,12], systolic Givens rotation [6,12,14,65], systolic modified Givens rotation [14,11,65,70], systolic-like modified Gram-Schmidt [68], and systolic block Householder transformation [57,59] have been reported elsewhere in the literature.

Recently, the problem of designing algorithms based on the Householder transformation (HT) and its associated systolic architectures has been of great interest [36,37,57,59,62,64]. This is because the HT generally outperforms the Givens rotation under finite precision computations [37,57,59,62]. It is also true that the Householder method requires less computation than the Givens and modified Gram-Schmidt methods do. Recently, a systolic architecture for the recursive block Householder transformation has been presented in [57,59]. Compared to the recursive block HT, the developed programmable complex HT systolic architecture in this paper saves $(M-1) \times N$ in computation time to form the upper triangular matrix during the initialization procedure, where

N is the number of sensors and M is the block size. Furthermore, rather than employing the real HT reported in [57,59], the complex HT is considered and developed since in many signal processing areas we often deal with complex data. Therefore, it is very important to develop a programmable complex HT systolic architecture for real-time high throughput modern signal processing applications.

The QR approaches for recursive least-squares (RLS) problem have played an important role in adaptive signal processing, such as adaptive beamforming, adaptive equalization, adaptive spectrum estimation, and so on. One of the reasons is that the apparent robust numerical stability is observed in those implementations since the rounding error caused by finite word length effect will not be accumulated by using the QR-RLS approaches. Another reason is that the QR-RLS algorithms can be mapped onto systolic arrays which are promising candidates for real-time signal processing applications and VLSI implementations. Basically, there are three approaches to the QR-RLS problem, namely, Householder transformations, Givens rotations, and the modified Gram-Schmidt method. Especially, the Givens rotation is a special case of the Householder transformation [55].

Systolic array implementations for QR-RLS algorithms have been explored by numerous researchers [6,4,9,24,36,37,57,59,65,66,68,69]. Gentleman and Kung introduced the linear least squares problem based on the Givens ro-

tations with systolic array implementation to derive the optimal weights [6]. Their method for linear least squares computation consists of two steps which are the orthogonal triangularization and the backward substitution. Step one was carried out by a triangular systolic array for the QR decomposition and step two was implemented by a systolic linear array for the back substitution. However, the triangular systolic array runs from the upper-left corner to the lower-right corner of the array, while the back solve systolic array runs in precisely the opposite direction. Although Gentleman-Kung's systolic architecture is not fully pipelined between the two modules, it was the pioneering work in discovering the systolic array implementation for the least squares problem. McWhirter then showed how the proposed Givens rotations based recursive least squares algorithm can be implemented efficiently on a single systolic array for directly computing the residual by avoiding the back solve processor [24]. The resulting McWhirter's Givens based-RLS array is both fully parallel and pipelined. Ling et al. proposed a recursive modified Gram-Schmidt (RMGS) algorithm for least-squares estimation [69]. Ling's RMGS-LS algorithm and architecture has also been demonstrated to be fully pipelined for computing the residual. In [70], Kalson and Yao have similar results.

Up to now, most of the systolic array implementations for the QR-RLS algorithms are based on the Givens rotation method and modified Gram-Schmidt method except the recently introduced systolic array for the block

householder transformation based RLS (SBHT-RLS) [57,59]. Compared to the SBHT-RLS method, the systolic architecture for the complex Householder transformation based recursive least squares (CHT-RLS) algorithm developed in this paper saves $(N-1)M$ in computation time to obtain the first residual vector, where N is the number of sensors and M is the block size. More precisely, the residuals can be obtained immediately from our CHT-RLS systolic architecture when an $N \times N$ data block is received by the sensors, while an $NM \times N$ data block is needed in [57,59]. In our proposed systolic architecture, the number of data snapshots needed for the initialization and the block size M for the recursive updating are controlled by simply sending the control code into the boundary cells and it can be changed freely by sending another control code into the cells.

This chapter is organized as follows. The first part consists of a complex Householder transformation algorithm and its associated systolic array processor [3]. We begin with the development of a systolic complex Householder algorithm which is programmable to handle both the initialization and recursive computation. The complex Householder algorithm requires N snapshots of data for the initialization to compute the upper triangular matrix while the recursive Householder algorithm in [57,59] needs $N \times M$ snapshots of data. Then we introduce the systolic architectures for the parallel complex Householder algorithms. A two-level pipelined implementation of the com-

plex Householder transformation is also considered. The second part consists of a complex Householder-based recursive least squares (CHT-RLS) systolic algorithm and its systolic array implementation. First, a systolic algorithm for CHT-RLS with fast initialization is described. Then, the systolic array processor of CHT-RLS systolic algorithm is proposed.

5.2 Systolic Arrays of Complex Householder Transformation

In many signal processing applications, the QR decomposition can provide a numerically stable and efficient solution. There are many schemes to perform such a decomposition, e.g. Givens/modified Givens, Householder, Gram-Schmidt/modified Gram-Schmidt. Of particular interest in this paper is a sample-by-sample form of the Householder orthogonalization technique. Since in many applications of signal processing, the observed data matrix is complex, it is necessary to consider the complex case of the Householder transformation. We assume X is a observed complex data matrix and let K be the number of snapshots and N is the number of sensors. The initialization is needed for k less than or equal to N since the upper triangular matrix is still not available and the recursive computation can be started when there are more than N data snapshots.

5.2.1 Systolic Complex Householder Algorithm

The following Lemma shows how a Householder transformation be applied to a column vector $\underline{x}$ to zero out all elements except the first one.

Lemma [55]

Suppose $\underline{x} \in \mathbf{C}^K$ and that $x_1 = |x_1|e^{j\theta}$ with $\theta \in \mathbf{R}$. Assume $\underline{x} \neq 0$ and define $\underline{u} = \underline{x} + e^{j\theta}||\underline{x}||_2\underline{e_1}$ where $\underline{e_1}^T = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$. Then the K by K complex Householder transformation B defined as

$$B = I - \frac{2}{\underline{u}^H\underline{u}}\underline{u}\underline{u}^H$$

is unitary and $B\underline{x} = -e^{j\theta}||\underline{x}||_2\underline{e_1}$ where H is complex conjugate transpose.

Initialization

The factorization of a data matrix $X \in \mathbf{C}^{N\times N}$ can be achieved by a sequence of Householder transformations [57,59] which produces a unitary N by N matrix Q and an upper triangular matrix R such that

$$X = Q^H \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $X = \begin{bmatrix} \underline{x}_1 & \underline{x}_2 & \cdots & \underline{x}_N \end{bmatrix}$. The algorithm for applying successive Householder transformations to zero out a given N by N complex observed data matrix X can be described as follows. Let

$$Q = Q_{N-1}\cdots Q_2Q_1 \tag{5.1}$$

be a sequence of Householder transformations applied to X where Q_i is an N by N complex Householder Transformation matrix of the form

$$Q_i = \begin{bmatrix} I_{i-1 \times i-1} & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & B_i \end{bmatrix}, \qquad for \;\; i = 1, \cdots, N-1, \tag{5.2}$$

where $B_i \in \mathbf{C}^{(N-i+1)\times(N-i+1)}$ is a unitary matrix given by [55]

$$B_i = I - \frac{2}{\underline{u}_i^H \underline{u}_i} \underline{u}_i \underline{u}_i^H, \tag{5.3}$$

and $\underline{u}_i$ is defined as

$$\underline{u}_i^T = \begin{bmatrix} x_i(t_i) + e^{j\theta_i(t_i)} ||\underline{x}_i||_2 & x_i(t_{i+1}) & \cdots & x_i(t_N) \end{bmatrix} \tag{5.4}$$

for $\underline{x}_i^T = \begin{bmatrix} x_i(t_i) & \cdots & x_i(t_N) \end{bmatrix}$ and the phase $\theta_i(t_i)$ is given by

$$\theta_i(t_i) = -j \log_e \frac{x_i(t_i)}{|x_i(t_i)|}.$$

As a result, applying a sequence of Householder transformations Q_i to the data matrix X can be described in two parts. First, for each $i = 1, \cdots, N-1$, we apply a Householder transformation to $\underline{x}_i$ and obtain

$$(B_i \underline{x}_i)^T = \begin{bmatrix} r_{ii} & 0 & \cdots & 0 \end{bmatrix}$$
$$for \;\; i = 1, 2, \cdots, N-1 \tag{5.5}$$

where $r_{ii} = -e^{j\theta_i(t_i)} ||\underline{x}_i||_2$.

Define a scalar parameter λ as

$$\lambda = \frac{2}{\underline{u}_i^H \underline{u}_i} = (||\underline{x}_i||_2(||\underline{x}_i||_2 + |x_i(t_i)|))^{-1}. \tag{5.6}$$

Then, the same Householder algorithm B_i is applied to the remaining $N-i$ column vectors $\underline{x}_k^T = \begin{bmatrix} x_k(t_i) & \cdots & x_k(t_N) \end{bmatrix} \in \mathbf{C}^{N-i+1}$, for $k = i+1, \cdots, N$. Thus, the new set of column vectors can be obtained as follows.

$$B_i \underline{x}_k = \underline{x}_k - \lambda \underline{u}_i \underline{u}_i^H \underline{x}_k = \underline{x}_k - \alpha \underline{u}_i, \tag{5.7}$$

where α is a scalar parameter given by

$$\alpha = \lambda \underline{u}_i^H \underline{x}_k = \lambda(\underline{x}_i^H \underline{x}_k + \underline{x}_k(t_i) e^{-j\theta_i(t_i)} ||\underline{x}_i||_2),$$

and

$$B_i \underline{x}_k = \begin{bmatrix} x_k(t_i) - \alpha x_i(t_i) + \alpha r_{ii} \\ x_k(t_{i+1}) - \alpha x_i(t_{i+1}) \\ \vdots \\ x_k(t_N) - \alpha x_i(t_N) \end{bmatrix}, \qquad for \;\; k = i+1, \cdots, N. \tag{5.8}$$

According to the above procedure, after applying Q_1 to X, the first column of $Q_1 X$ is zeroed except for the first element. The second column of $Q_2 Q_1 X$ is zeroed from the third component to the $M-th$ when a chosen $N-1$ by $N-1$ unitary matrix H_2 is applied to the $N-1$ by $N-1$ lower right submatrix of $Q_1 X$. It is obvious that $Q_2 Q_1 X$ has zeros below the diagonal in both the first two columns. Continuing in this way, the data matrix X can be transformed

into a upper triangular form by applying $N-1$ unitary transformations to it. It is well known that the number of arithmetic operations gradually decreases in each of the subsequent Householder transformations.

Recursive Complex Householder Algorithm

The triangular matrix R can be updated by employing unitary Householder transformations P which has the form

$$P \begin{bmatrix} \beta R \\ X \end{bmatrix} = \begin{bmatrix} R' \\ 0 \end{bmatrix}, \tag{5.9}$$

where $R = \begin{bmatrix} r_{11} & r_{21} & \cdots & r_{N1} \\ 0 & r_{22} & \cdots & r_{N2} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & r_{NN} \end{bmatrix}$, $X = \begin{bmatrix} \underline{x}_1 & \underline{x}_2 & \cdots & \underline{x}_N \end{bmatrix}$, P^H represents a sequence of complex Householder transformations used to zero out the new complex M by N data matrix X, and β is the forgetting factor.

The sequence of Householder transformations to update the triangular matrix R is given by

$$P = P_N P_{N-1} \cdots P_2 P_1, \tag{5.10}$$

where an $M+N$ by $M+N$ complex Householder transformation P_i has the

form

$$P_i = \begin{bmatrix} I_{i-1\times i-1} & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & H_i \end{bmatrix}, \qquad for \quad i = 1, 2, \cdots, N, \tag{5.11}$$

where $B_i \in \mathbf{C}^{(M+N-i+1)\times(M+N-i+1)}$ is a unitary matrix given by

$$B_i = I - \frac{2}{\underline{u}_i^H \underline{u}_i} \underline{u}_i \underline{u}_i^H. \tag{5.12}$$

When given a column vector $\underline{x}_i^{'}$ of the form

$$\underline{x}_i^{'T} = \begin{bmatrix} \beta r_{ii} & 0 & \cdots & 0 & x_i(t_1) & \cdots & x_i(t_M) \end{bmatrix}, \tag{5.13}$$

$\underline{u}_i$ can be defined by

$$\underline{u}_i^T = \begin{bmatrix} \beta r_{ii} + e^{j\theta_{r_{ii}}} ||\underline{x}_i^{'}||_2 & 0 & \cdots & 0 & x_i(t_1) & \cdots & x_i(t_M) \end{bmatrix}, \tag{5.14}$$

where $\theta_{r_{ii}}$ is a real parameter given by $\theta_{r_{ii}} = -j \log_e \frac{r_{ii}}{|r_{ii}|}$.

Therefore, the Householder transformation B_i is readily available as

$$B_i = \begin{bmatrix} H_{1\times 1} & \vdots & 0 & \vdots & H_{M\times 1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \vdots & I_{N-i\times N-i} & \vdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ H_{1\times M} & \vdots & 0 & \vdots & H_{M\times M} \end{bmatrix}, \tag{5.15}$$

where $H_{1\times 1} = 1 - \lambda(\beta^2 r_{ii}^* r_{ii} + ||\underline{x}_i^{'}||_2^2 + 2\beta|r_{ii}|||\underline{x}_i^{'}||_2)$,

$$H_{M\times 1} = -\lambda(\beta r_{ii} + e^{j\theta_{r_{ii}}}||\underline{x}_i^{'}||_2)\underline{x}_i^H,$$

$$H_{1\times M} = -\lambda(\beta r_{ii}^{*} + e^{-j\theta_{r_{ii}}}||\underline{x}_i^{'}||_2)\underline{x}_i,$$

$$H_{M\times M} = I - \lambda \underline{x}_i \underline{x}_i^H,$$

and

$$\lambda = \frac{2}{\underline{u}_i^H \underline{u}_i} = (||\underline{x}_i^{'}||_2(||\underline{x}_i^{'}||_2 + \beta|r_{ii}|))^{-1}. \tag{5.16}$$

Given a column vector $\underline{x}_i^{'}$ described in equation (13), a Householder transformation applied to this matrix gives

$$(B_i\underline{x}_i^{'})^T = \begin{bmatrix} r_{ii}^{'} & 0 & \cdots & 0 \end{bmatrix}$$
$$for \quad i = 1, 2, \cdots, N \tag{5.17}$$

where $r_{ii}^{'} = -e^{j\theta_{r_{ii}}}||\underline{x}_i^{'}||_2$.

When the same Householder transformation is applied to the rest of column vectors $\underline{x}_k^{'}$ with

$$\underline{x}_k^{'T} = \begin{bmatrix} \beta r_{ki} & \cdots & \beta r_{kk} & 0 & \cdots & 0 & x_k(t_1) & \cdots & x_k(t_M) \end{bmatrix},$$

the new set of column vectors can by obtained by

$$B_i \underline{x}_k^{'} = \begin{bmatrix} \beta r_{ki} - \alpha r_{ii} + \alpha r_{ii}^{'} \\ \beta r_{k\,i+1} \\ \vdots \\ \beta r_{kk} \\ 0 \\ \vdots \\ 0 \\ x_k(t_1) - \alpha x_i(t_1) \\ \vdots \\ x_k(t_M) - \alpha x_i(t_M) \end{bmatrix}, \quad for \quad k = i+1, \cdots, N, \tag{5.18}$$

where α is given by $\alpha = \lambda \underline{u}_i^H \underline{x}_k^{'} = \lambda(\underline{x}_i^{'H} \underline{x}_k^{'} - r_{ii}^{'*} r_{ki})$.

It is apparent that the upper triangular matrix can be updated by applying a sequence of N Householder transformations. It is also known that the number of data to be zeroed is the same for each of the N Householder transformations at updating procedure while the number of data to be zeroed is reduced at initialization.

5.2.2 VLSI Array Processors Implementation

The parallel complex Householder algorithms with initialization have been presented so far. We now consider the issue of VLSI systolic implementation.

Figure 5.1 shows the boundary cell and internal cell of the systolic recursive Householder transformations. The operation of the boundary cell is given in Table 5.1 and that of the internal cell is described in Table 5.2. Based on the initialization procedure described before, the block size is decreasing such that only N data rows are needed to obtain the first upper triangular matrix R. For the systolic array implementation, a control code to change the block size is sent down to the boundary cells as illustrated in Figure 5.3. In [57,59], a fixed block size is used instead. As a result, $N \times M$ data rows are used for the initialization. Figure 5.2 illustrates the operation carried out by a given pair of boundary and internal cells. The data flowed from the boundary cell to internal cell is pipelined sample-by-sample. The boundary cell and internal cell are functioned as follows. In the boundary cell, the previous data r stored in the processor element is sent to the internal cell first, then the input data x received is accumulated and added, and also sent to the internal cell. According to Table 5.1, the newly updated data r which replaces the previous r is also sent to the internal cell. The data flow and computation in both processor elements are fully pipelined down to the word level to update the previous data and to produce the output data y to the next cell. A similar two-level pipelined architecture for the modified Householder transformation algorithm [60] is given in [59]. The two-level pipelined architectures for the conventional and modified Householder transformation have some similarities and differ-

ences. Compared to the two-level pipelined architecture shown in [59], the architecture based on conventional Householder transformation described in Figure 5.2 requires one more multiplication and addition computation than that of the modify Householder transformation. The similarity of two architectures is that both are fully pipelinable at the vector and word levels. The triangular systolic architecture for the parallel complex Householder algorithm is shown in Figure 5.3. In adaptive antenna and radar applications, the period of updating the optimum weights is significantly larger than the actual computation time [68], and the two-dimensional systolic array processors can be reduced into one-dimensional systolic array processors by employing a simple feedback configuration as illustrated in Figure 5.4. The one-dimensional linear architecture using the feedback configuration and the two-dimensional triangular architecture require their own local memory and some minimal control circuity and programmable capabilities.

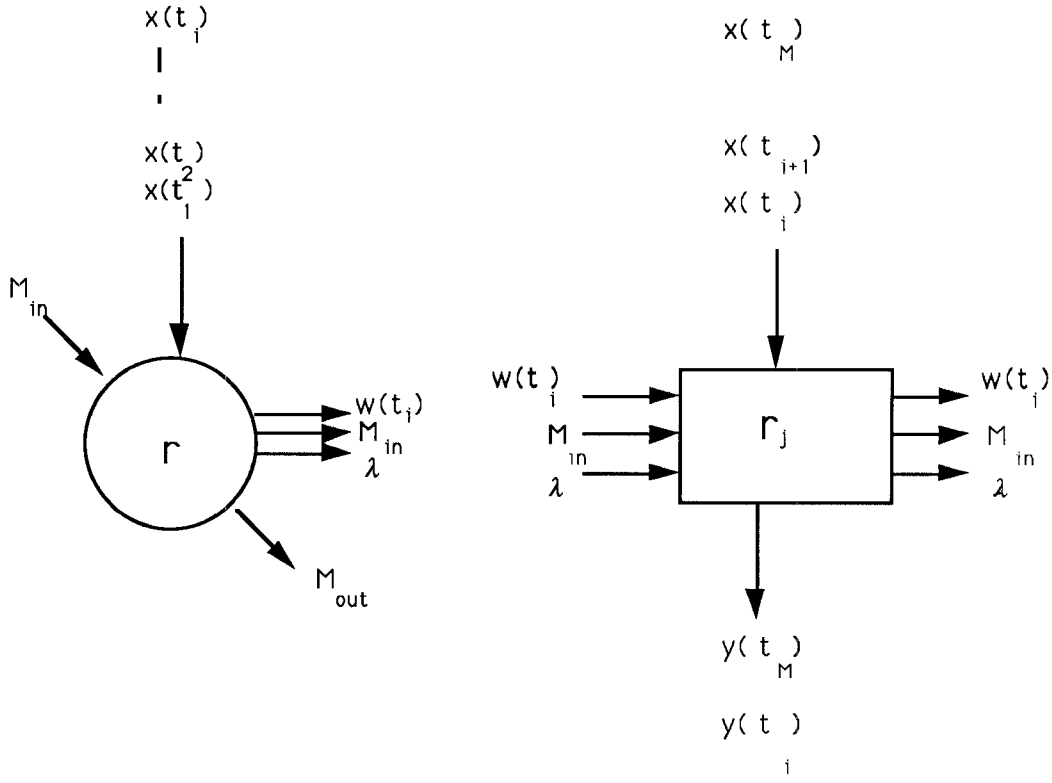


Figure 5.1: The Boundary and Internal Cells of Systolic Householder Transformation

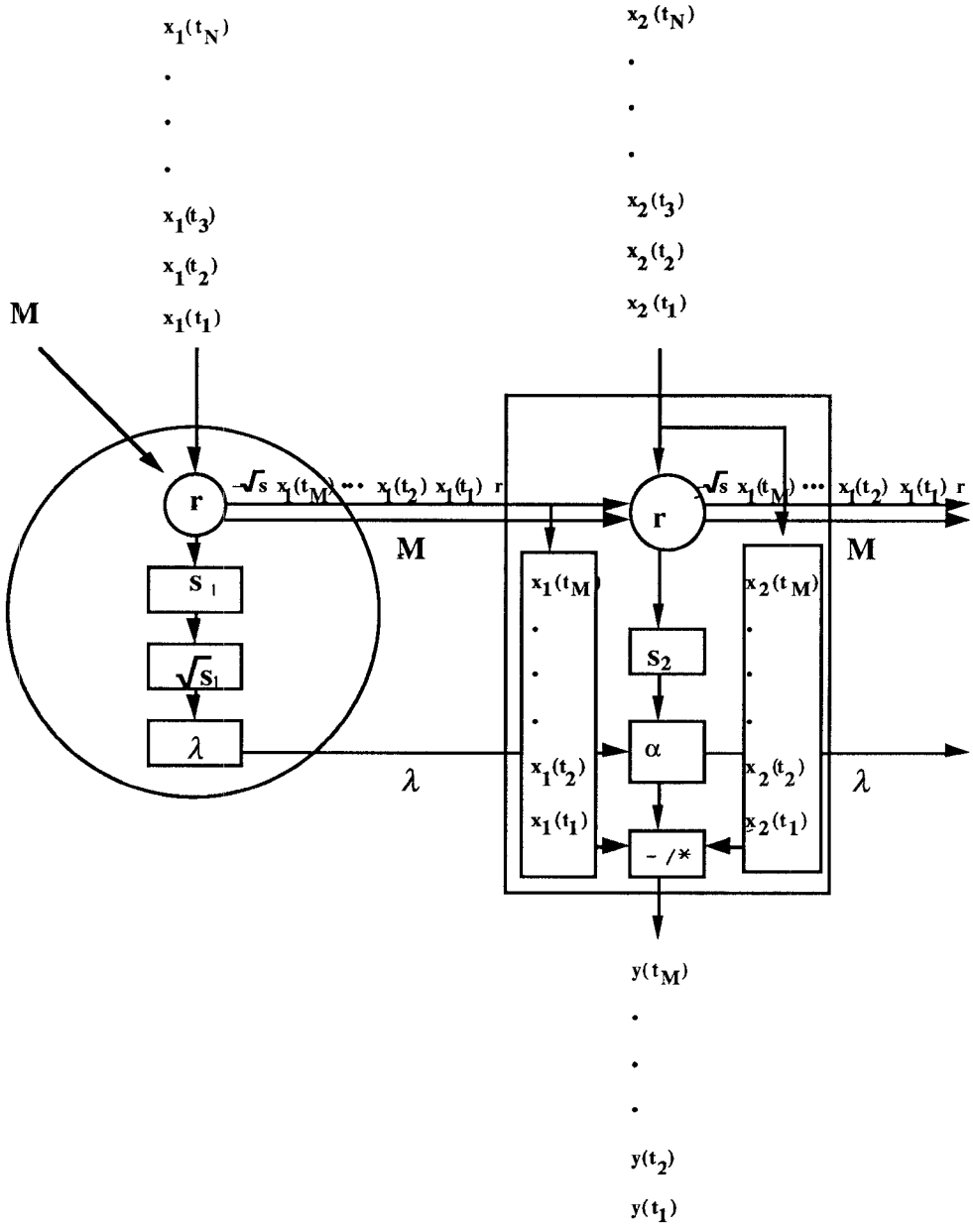


Figure 5.2: Processor Pair of Boundary and Internal Cells

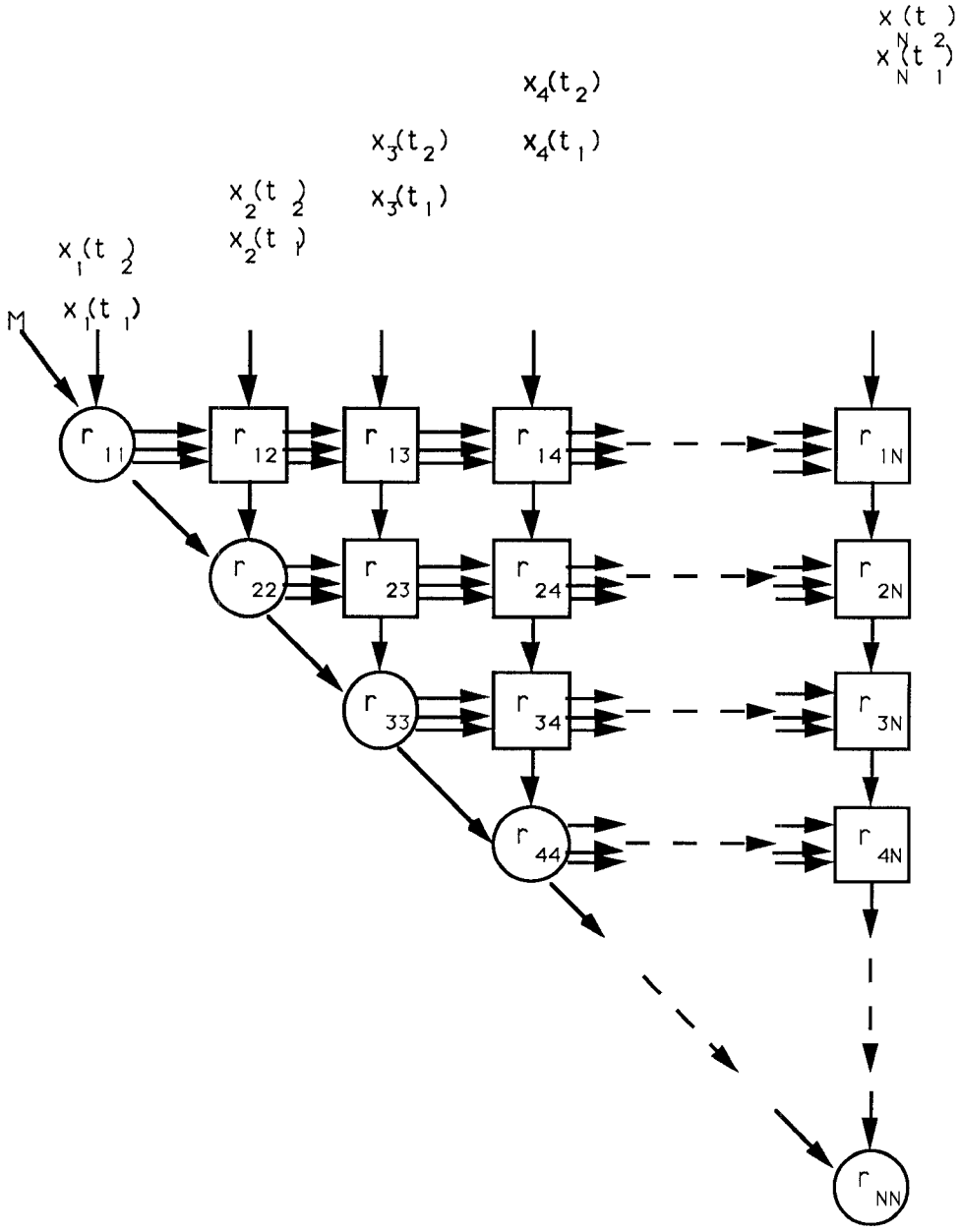


Figure 5.3: Triangular Systolic Architecture for Householder Transformation

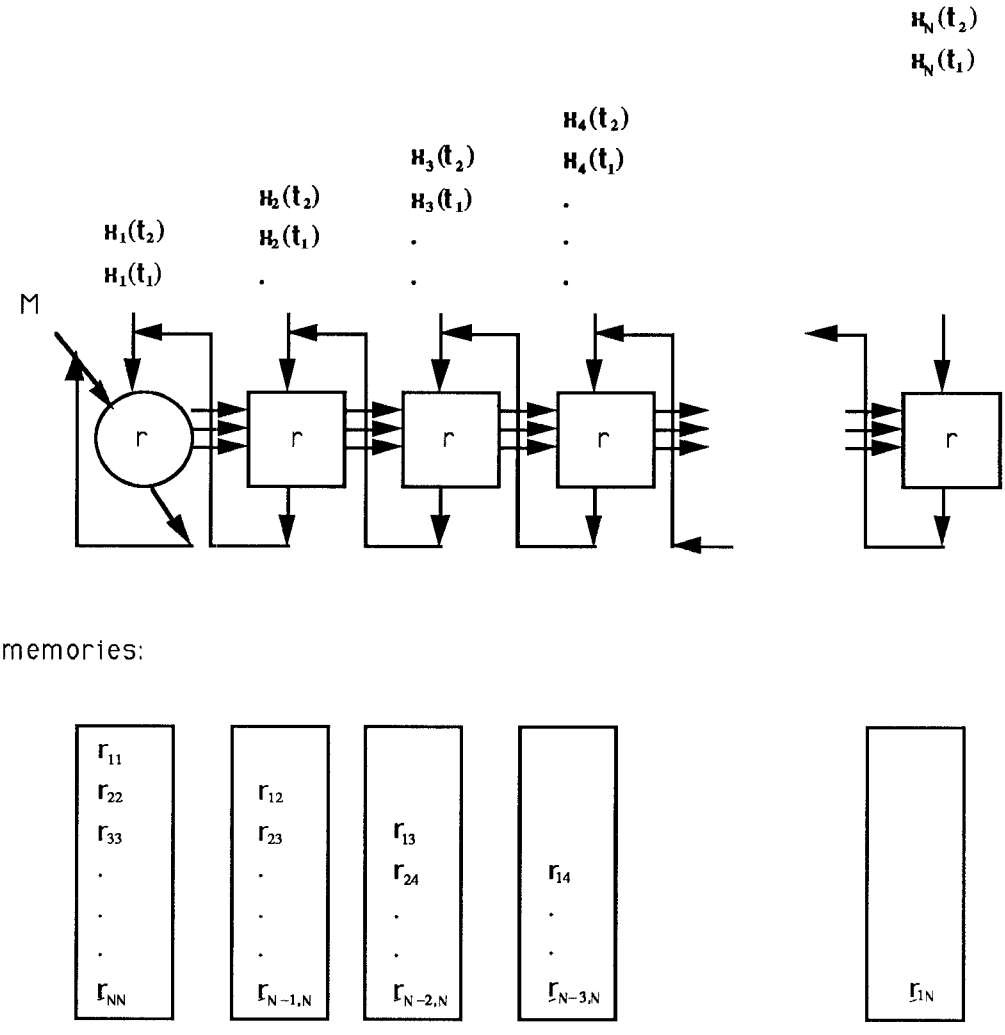


Figure 5.4: Linear Triangular Systolic Architecture for Householder Transformations with Feedback Configuration

Initialization :	*Recursive* :
$r \leftarrow x(t_1)$	$e^{j\theta_r} = \frac{r}{\|r\|}$
$e^{j\theta_r} = \frac{r}{\|r\|}$	$s \leftarrow 0$
$s \leftarrow 0$	$for \;\; i = 0, M_{in}$
$for \;\; i = 1, M_{in}$	$w(t_i) \leftarrow r$
$w(t_i) \leftarrow r$	$s \leftarrow s + r^* r$
$s \leftarrow s + r^* r$	$r \leftarrow x(t_{i+1})$
$r \leftarrow x(t_{i+1})$	end
end	$\sigma \leftarrow \sqrt{s}$
$\sigma \leftarrow \sqrt{s}$	$r \leftarrow -e^{j\theta_r}\sigma$
$r \leftarrow -e^{j\theta_r}\sigma$	$\lambda \leftarrow \sigma(\sigma + \beta\|r\|)$
$\lambda \leftarrow \sigma(\sigma + \|r\|)$	$w(t_{M_{in}+1}) \leftarrow r$
$w(t_{M_{in}+1}) \leftarrow r$	$M_{out} \leftarrow M_{in}$
$M_{out} \leftarrow M_{in} - 1$	

Table 5.1: The Algorithm for the Boundary Cell of Complex Householder Transformation

Initialization :	*Recursive* :
$x(t_{M_{in}+1}) \leftarrow -x(t_1)$	$x(t_{M_{in}+1}) \leftarrow -r$
$s \leftarrow 0$	$s \leftarrow 0$
$for \;\; i = 1, M_{in} + 1$	$for \;\; i = 0, M_{in} + 1$
$s \leftarrow s + w^*(t_i)x(t_i)$	$s \leftarrow s + w^*(t_i)r$
end	$r \leftarrow x(t_{i+1})$
$\alpha \leftarrow \frac{s}{\lambda}$	end
$r \leftarrow -x(t_{M_{in}+1}) - \alpha w(t_1) + \alpha w(t_{M_{in}+1})$	$\alpha \leftarrow \frac{s}{\lambda}$
$for \;\; i = 2, M$	$r \leftarrow -\beta x(t_{M_{in}+1}) - \alpha w(t_0) + \alpha w(t_{M_{in}+1})$
$y(t_i) \leftarrow x(t_i) - \alpha w(t_i)$	$for \;\; i = 1, M$
end	$y(t_i) \leftarrow x(t_i) - \alpha w(t_i)$
	end

Table 5.2: The Algorithm for the Internal Cell of Complex Householder Transformation

5.3 Systolic CHT-RLS Algorithm and Architecture

In this section we consider the recursive least-squares based on the systolic complex Householder transformations (CHT-LS) derived above.

Assume the observed data and the desired data are received by $N+1$ sensors of an adaptive array system for time period of m snapshots. Then, the $k \times N$ observed data matrix and the $k \times 1$ desired data vector are

$$X(1:k) = \begin{bmatrix} \underline{x}^T(t_1) \\ \underline{x}^T(t_2) \\ \vdots \\ \underline{x}^T(t_k) \end{bmatrix} \tag{5.19}$$

and

$$\underline{y}(1:k) = \begin{bmatrix} y(t_1) \\ y(t_2) \\ \vdots \\ y(t_k) \end{bmatrix}. \tag{5.20}$$

The least-squares residual vector is

$$\underline{e}(1:k) = X(1:k)\underline{w}(k) - \underline{y}(1:k) \tag{5.21}$$

where the weights for each of N sensors are

$$\underline{w}(k) = \begin{bmatrix} w_1(t_k) \\ w_2(t_k) \\ \vdots \\ w_N(t_k) \end{bmatrix} \tag{5.22}$$

and the residual vector is

$$\underline{e}(1:k)^T = \begin{bmatrix} e(t_1) & e(t_2) & \cdots & e(t_k) \end{bmatrix}.$$

The optimal weight vector $\underline{w}_{opt}(k)$ minimizes the quantity

$$E(1:k) = ||X(1:k)\underline{w}(k) - \underline{y}(1:k)||_2. \tag{5.23}$$

The solution to this minimization problem is

$$\underline{w}_{opt}(k) = (X^H(1:k)X(1:k))^{-1}X^H(1:k)\underline{y}(1:k). \tag{5.24}$$

5.3.1 Systolic CHT-RLS Algorithm

The approach described in (5.24) is generally known as direct sample matrix inversion (SMI). It is well known that the classical method, the sample matrix inversion method (SMI), may sometimes lead to undesired numerical characteristics due to ill-conditioned matrices. This means that an extremely high arithmetic precision is required when employing the sample matrix inversion method. To alleviate such roundoff sensitivity caused by the SMI method,

the QR decomposition deserves serious consideration. A family of algorithms based on the QR decomposition can be employed. These include the Givens, modified Givens, Householder and modified Gram-Schmidt orthogonalization techniques. In this section, we introduce complex Householder transformation based-least squares (CHT-LS) algorithm to compute the residual. In CHT-LS, as before, the initialization is performed when k, the number of data snapshots, is less than or equal to N, and the recursive computation is started when there are more than N data snapshots.

Initialization Procedure

The initialization procedure for the complex Householder transformation to form the upper triangular matrix requires only N data snapshots. The factorization of a data matrix $X(1:N) \in \mathbf{C}^{N\times N}$ described in the last section can be achieved by a sequence of Householder transformations [57,59] and produces a unitary N by N matrix $Q(N)$ and an $N \times N$ upper triangular matrix $R(N)$ such that

$$X(1:N) = Q^H(N) \begin{bmatrix} R(N) \\ \\ 0 \end{bmatrix},$$

where $X(1:N) = \begin{bmatrix} \underline{x}^T(t_1) & \underline{x}^T(t_2) & \cdots & \underline{x}^T(t_N) \end{bmatrix}$.

The least-squares residual vector for the initialization procedure is

$$\underline{e}(1:N) = X(1:N)\underline{w}(N) - \underline{y}(1:N). \tag{5.25}$$

Therefore,

$$\begin{aligned} E(1:N) &= ||Q(N)X(1:N)\underline{w}(N) - Q(N)\underline{y}(1:N)||_2 \\ &= ||R(N)\underline{w}(N) - \underline{u}(N)||_2. \end{aligned} \tag{5.26}$$

Since $Q(N)$ is unitary, according to (5.26), the optimal weight vector is given by

$$R(N)\underline{w}_{opt}(N) - \underline{u}(N) = 0 \tag{5.27}$$

Substituting (5.27) into (5.25), the least-squares residual vector for the initialization is given by

$$\begin{aligned} \underline{e}(1:N) &= X(1:N)\underline{w}_{opt}(N) - \underline{y}(1:N) \\ &= Q^H(N)R(N)R^{-1}(N)\underline{u}(N) - Q^H(N)\underline{u}(N) \\ &= \underline{0}. \end{aligned} \tag{5.28}$$

Recursive Updating

Given a $k \times N$ matrix $X(1:k)$, where $k > N$, a $k \times k$ unitary matrix $Q(k)$ can be generated such that

$$X(1:k) = Q^H(k) \begin{bmatrix} R(k) \\ \\ 0 \end{bmatrix} \tag{5.29}$$

where $R(k)$ is a $N \times N$ upper triangular matrix.

The least-squares residual vector is

$$\underline{e}(1:k) = X(1:k)\underline{w}(k) - \underline{y}(1:k) \tag{5.30}$$

The CHT-RLS algorithm is formulated to derive the optimal weight vector by minimizing the following quantity

$$\begin{aligned} E(1:k) &= ||Q(k)X(1:k)\underline{w}(k) - Q(k)\underline{y}(1:k)||_2 \\ &= ||\begin{bmatrix} R(k) \\ 0 \end{bmatrix}\underline{w}(k) - \begin{bmatrix} \underline{u}(k) \\ \underline{v}(k) \end{bmatrix}||_2 \end{aligned} \tag{5.31}$$

where $Q(k)$ can be partitioned into $Q_1(k)$, a $N \times k$ matrix, and $Q_2(k)$, a $(k-N) \times k$ matrix, is

$$Q(k) = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{5.32}$$

and $\underline{u}(k)$, a $N \times 1$ vector, and $\underline{v}(k)$, a $(k-N) \times 1$, vector are defined by

$$Q_1\underline{y}(1:k) = \underline{u}(k), \tag{5.33}$$

$$Q_2\underline{y}(1:k) = \underline{v}(k). \tag{5.34}$$

To minimize the quantity $E(1:k)$, the optimal weight vector obtained from (5.31) has the form

$$R(k)\underline{w}_{opt}(k) = \underline{u}(k), \tag{5.35}$$

where the optimal weight vector is given by

$$\underline{w}_{opt}(k) = \begin{bmatrix} w_1(t_k) \\ w_2(t_k) \\ \vdots \\ w_N(t_k) \end{bmatrix},$$

Then, substituting (5.35) into (5.31), the residual vector is given by

$$\begin{aligned}\hat{e}(1:k) &= Q^H(k)\begin{bmatrix} R(k) \\ 0 \end{bmatrix}\underline{w}_{opt}(k) - Q^H(k)\begin{bmatrix} u(k) \\ v(k) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ -Q_2 v(k) \end{bmatrix}. \end{aligned} \tag{5.36}$$

Therefore, the residual vector for the least-squares problem is

$$\hat{e}((N+1):k) = -Q_2^H v(k). \tag{5.37}$$

When a new $M \times N$ data block is received by an adaptive array system, the observed data matrix becomes

$$X(1:(k+M)) = \begin{bmatrix} \beta X(1:k) \\ X((k+1):(k+M)) \end{bmatrix}, \tag{5.38}$$

where the $M \times N$ data matrix is given by

$$X((k+1):(k+M)) = \begin{bmatrix} \underline{x}^T(t_{k+1}) \\ \underline{x}^T(t_{k+2}) \\ \vdots \\ \underline{x}^T(t_{k+M}) \end{bmatrix},$$

and the desired data vector is given by

$$\underline{y}(1:(k+M)) = \begin{bmatrix} \underline{y}(1:k) \\ \underline{y}((k+1):(k+M)) \end{bmatrix}, \tag{5.39}$$

where $\underline{y}((k+1):(k+M))$ is a new desired data vector given by

$$\underline{y}((k+1):(k+M)) = \begin{bmatrix} y(t_{k+1}) \\ y(t_{k+2}) \\ \vdots \\ y(t_{k+M}) \end{bmatrix}.$$

The new $(k+M)\times(k+M)$ complex Householder transformation $Q(k+M)$ is defined as

$$Q(k+M) = P(k+M)\overline{Q}(k) \tag{5.40}$$

where $\overline{Q}(k)$ is given by

$$\overline{Q}(k) = \begin{bmatrix} Q(k) & 0 \\ 0 & I_{M\times M} \end{bmatrix}.$$

The desired Householder transformation $P(k+M)$ described in [57,59] for updating the upper triangular matrix has the form

$$P(k+M) = \begin{bmatrix} H_{11}(k+M) & 0 & H_{12}(k+M) \\ 0 & I_{(k-N)\times(k-N)} & 0 \\ H_{21}(k+M) & 0 & H_{22}(k+M) \end{bmatrix}, \tag{5.41}$$

where $H_{11}(k+M)$ is an $N \times N$ matrix,

$H_{12}(k+M)$ is an $N \times M$ matrix,

$H_{21}(k+M)$ is an $M \times N$ matrix,

and $H_{22}(k+M)$ is an $M \times M$ matrix.

Applying $\overline{Q}(k) \in \mathbf{C}^{(N+M)\times(N+M)}$ to the $(k+M) \times N$ data matrix $X(1 : (k+M))$, gives

$$\begin{aligned}
\overline{Q}(k)X(1:(k+M)) &= \begin{bmatrix} Q(k) & 0 \\ 0 & I_{M\times M} \end{bmatrix} \begin{bmatrix} \beta X(1:k) \\ X((K+1):(k+M)) \end{bmatrix} \\
&= \begin{bmatrix} \beta R(k) \\ 0 \\ X((k+1):(k+M)) \end{bmatrix}. \qquad (5.42)
\end{aligned}$$

Therefore, the $N \times N$ upper triangular matrix $R(k+M)$ can be updated by employing the new unitary Householder transformation $Q(k+M)$ which has the form

$$\begin{aligned}
Q(k+M)X(1:(k+M)) &= P(k+M) \begin{bmatrix} \beta R(k) \\ 0 \\ X((k+1):(k+M)) \end{bmatrix} \\
&= \begin{bmatrix} R(k+M) \\ 0 \\ 0 \end{bmatrix}, \qquad (5.43)
\end{aligned}$$

where $R(k+M) = H_{11}(k+M)\beta R(k) + H_{12}(k+M)X((k+1):(k+M))$.

The procedure for applying the updated complex Householder transformation to the desired data vector $\underline{y}(1 : (k+M))$ is the same as that described for the observed data matrix. First, applying $\overline{Q}(k)$ to the desired data vector

$\underline{y}(1:(k+M))$, we have

$$\overline{Q}(k)\underline{y}(1:(k+M)) = \begin{bmatrix} Q(k) & 0 \\ 0 & I_{M\times M} \end{bmatrix} \begin{bmatrix} \beta\underline{y}(1:k) \\ \underline{y}((k+1):(k+M)) \end{bmatrix}$$

$$= \begin{bmatrix} \beta\underline{u}(k) \\ \beta\underline{v}(k) \\ \underline{y}((k+1):(k+M)) \end{bmatrix}. \tag{5.44}$$

Then, by employing the new unitary Householder transformation $Q(k+M)$ to the desired data vector $\underline{y}(1:(k+M))$, we have

$$Q(k+M)\underline{y}(1:(k+M)) = P(k+M) \begin{bmatrix} \beta\underline{u}(k) \\ \beta\underline{v}(k) \\ \underline{y}((k+1):(k+M)) \end{bmatrix}$$

$$= \begin{bmatrix} \underline{u}(k+M) \\ \underline{v}(k+M) \end{bmatrix}, \tag{5.45}$$

where $\underline{u}(k+M) = H_{11}(k+M)\beta\underline{u}(k) + H_{12}(k+M)\underline{y}((k+1):(k+M))$,

$$\underline{v}(k+M) = \begin{bmatrix} \beta\underline{v}(k) \\ \alpha(k+M) \end{bmatrix},$$

and $\alpha(k+M) = H_{21}(k+M)\beta u(k) + H_{22}(k+M)\underline{y}((k+1):(k+M))$.

The residual vector for the recursive least-squares problem is

$$\underline{e}(1:(k+M)) = X(1:(k+M))\underline{w}(k+M) + \underline{y}(1:(k+M)), \tag{5.46}$$

and by the definition

$$\begin{aligned} E(1:(k+M)) &= ||Q(k+M)X(1:(k+M))\underline{w}(k+M) - Q(k+M)\underline{y}(1:k+M)||_2 \\ &= ||\begin{bmatrix} R(k+M) \\ \\ 0 \end{bmatrix}\underline{w}(k+M) - \begin{bmatrix} \underline{u}(k+M) \\ \\ \underline{v}(k+M) \end{bmatrix}||_2. \end{aligned} \tag{5.47}$$

To minimize the quantity $E(1:(k+M))$, the optimal weight vector obtained from the (5.47) is

$$R(k+M)\underline{w}_{opt}(k+M) = \underline{u}(k+M). \tag{5.48}$$

Then, substituting (5.48) into (5.47), the optimal residual vector becomes

$$\begin{aligned} \widehat{e}(1:(k+M)) &= Q^H(k+M)\begin{bmatrix} R(k+M) \\ \\ 0 \end{bmatrix}\underline{w}_{opt}(k+M) - Q^H(K+M)\begin{bmatrix} u(k+M) \\ \\ v(k+M) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \\ -Q_2^H(k+M)v(k+M) \end{bmatrix} \end{aligned} \tag{5.49}$$

Therefore, the most current optimal residual vector is

$$\begin{aligned} \widehat{e}((N+1):(k+M)) &= -Q_2^H(k+M)v(k+M) \\ &= -Q_2^H(k+M)\begin{bmatrix} \beta\underline{v}(k) \\ \\ \alpha(k+M) \end{bmatrix}. \end{aligned} \tag{5.50}$$

The new complex Householder transformation $Q(k+M)$ has the form

$$Q(k+M) = \begin{bmatrix} Q_1(k+M) \\ \\ Q_2(k+M) \end{bmatrix}$$

$$
\begin{aligned}
&= P(k+M)\overline{Q}(k) \\
&= \begin{bmatrix} H_{11}(k+M) & 0 & H_{12}(k+M) \\ 0 & I_{(k-N)\times(k-N)} & 0 \\ H_{21}(k+M) & 0 & H_{22}(k+M) \end{bmatrix} \begin{bmatrix} Q_1(k) & 0 \\ Q_2(k) & 0 \\ 0 & I_{M\times M} \end{bmatrix} \\
&= \begin{bmatrix} H_{11}(k+M)Q_1(k) & H_{12}(k+M) \\ Q_2(k) & 0 \\ H_{21}(k+M)Q_1(k) & H_{22}(k+M) \end{bmatrix}. \qquad (5.51)
\end{aligned}
$$

Therefore, $Q_2(k+M)$ has the form

$$
Q_2(k+M) = \begin{bmatrix} Q_2(k) & 0 \\ H_{21}(k+M)Q_1(k) & H_{22}(k+M) \end{bmatrix}. \qquad (5.52)
$$

Substituting (5.52) into (5.50). we have

$$
\begin{aligned}
\underline{\hat{e}}((N+1):(k+M)) &= \begin{bmatrix} \underline{e}((N+1):k) \\ \underline{e}((k+1):(k+M)) \end{bmatrix} \qquad (5.53) \\
&= -\begin{bmatrix} Q_2^H(k) & Q_1^H(k)H_{21}^H(k+M) \\ 0 & H_{22}^H(k+M) \end{bmatrix} \begin{bmatrix} \beta\underline{v}(k) \\ \alpha(k+M) \end{bmatrix} \\
&= -\begin{bmatrix} Q_2^H(k)\beta\underline{v}(k) + Q_1^H(k)H_{21}^H(k+M)\alpha(k+M) \\ H_{22}^H(k+M)\alpha(k+M) \end{bmatrix},
\end{aligned}
$$

where the optimal residual vector $\underline{\hat{e}}((k+1):(k+M))$ during the time period from t_{k+1} to t_{k+M} is given by

$$
\underline{\hat{e}}((k+1):(k+M)) = -H_{22}^H(k+M)\alpha(k+M), \qquad (5.54)
$$

where

$$H_{22}^{H} = \prod_{i=1}^{N} H_{M\times M}^{(i)},$$

where $H_{M\times M}^{(i)}$ is given in (5.15), i denotes the ith Householder transformation, and

$$\underline{e}((k+1):(k+M)) = \begin{bmatrix} e(t_{k+1}/(t_{k+1}:t_{k+M})) \\ e(t_{k+2}/(t_{k+1}:t_{k+M})) \\ \vdots \\ e(t_{k+M}/(t_{k+1}:t_{k+M})) \end{bmatrix}. \tag{5.56}$$

Compared to the SBHT-RLS algorithm in [59], our CHT-RLS algorithm saves $(N-1)M$ computation time in the initialization. Compared to McWhirter's GR-RLS algorithm, our algorithm has better estimation of the residuals since the CHT-RLS algorithm uses a data block to compute the residuals. Then, the more data information is used for CHT-RLS than GR-RLS to compute the residuals. For example, from (5.56), it is seen that the data block at time from t_{k+1} to t_{k+M} is used to derive the residual at each instant time.

5.3.2 Systolic Array Implementation

The parallel complex Householder transformation based-recursive least-squares algorithm with fast initialization has been presented so far. We now consider the issue of VLSI systolic implementation. The systolic architecture for the parallel complex Householder transformation based-recursive least-squares al-

gorithm is shown in Figure 5.5 which is able to obtain the residual immediately. However, it involves the matrix by matrix multiplication in the boundary cell between which a large bandwidth of transmission line is required. Therefore, in Figure 5.7 a backward propagation array is added into the Figure 5.5 to avoid the matrix-matrix multiplication and br replaced by vector-vector multiplication as pointed out in [59]. In Figure 5.7, the delayed buffers are needed for each row to temperately store $w(t_i)$ for vector multiplication. The boundary cell, internal cell, and the final cell are illustrated in Figure 5.6. The systolic algorithm for the boundary cell of the CHT-RLS system is shown in Table 5.3, the algorithm for the internal cell is shown in Table 5.4, and the algorithm for the final cell is described in Table 5.5.

5.4 Application to Array Processing

Adaptive arrays are currently the subject of extensive investigation for the application of automatically suppression of the sidelobe interference or jamming signals in many military radar, communications, and navigation systems. A more sophisticated adaptive array system is required to have rapid convergence, high cancellation performance and operational flexibility and is also necessitated to achieve high throughput rate and instant response in the applications of real-time signal processing. For those applications it is necessary to build an open-loop recursive QR decomposition-based systolic array system.

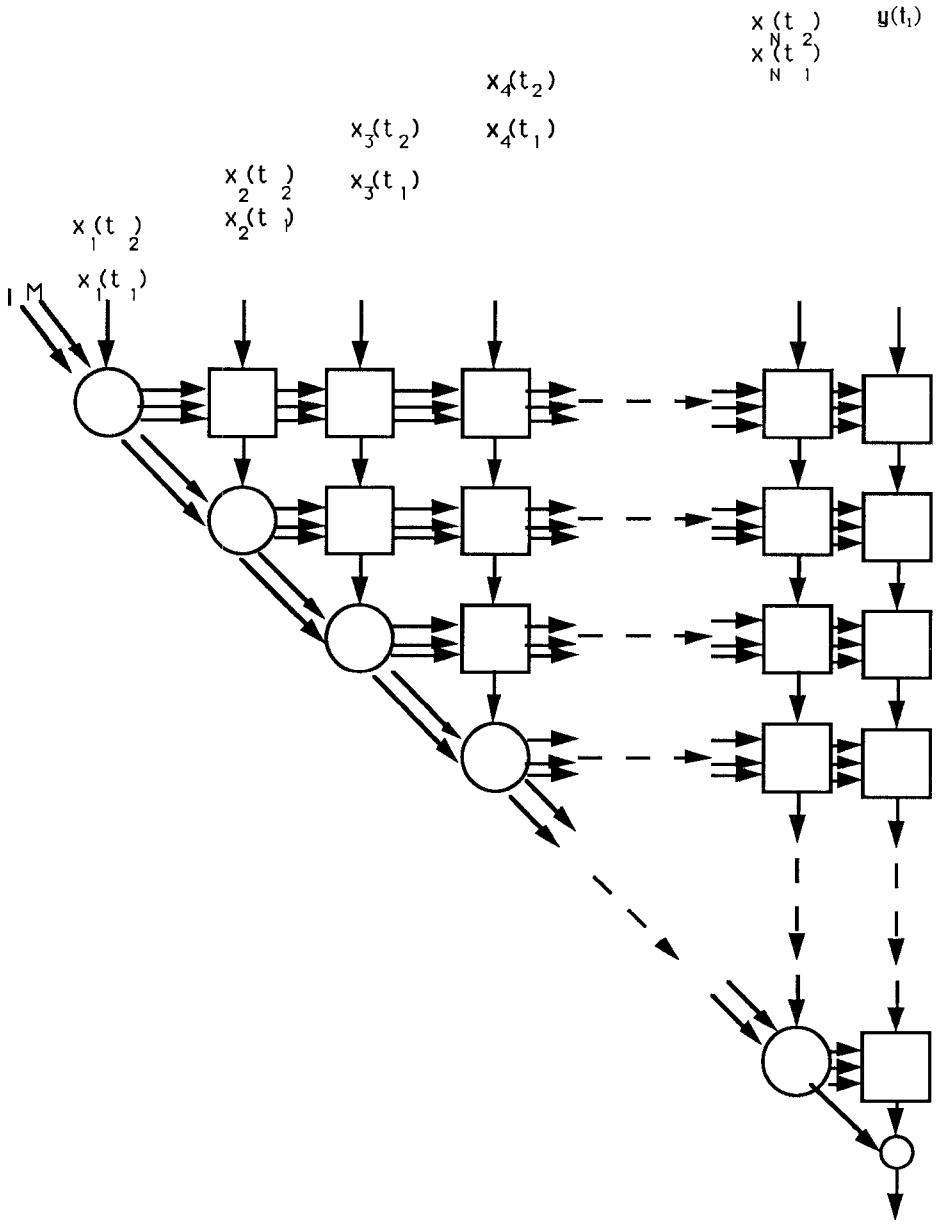


Figure 5.5: Systolic Architecture for CHT-RLS

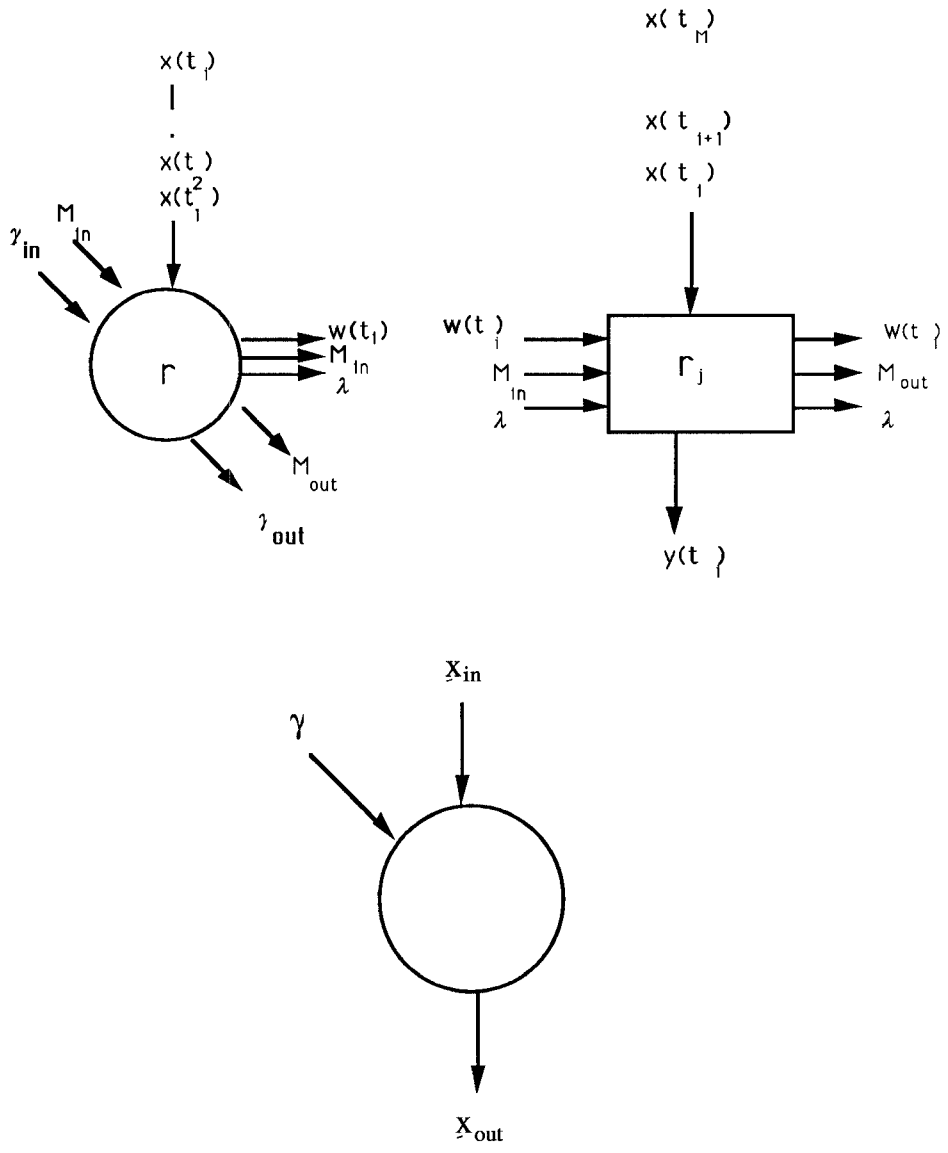


Figure 5.6: The Boundary, Internal, and Final Cells of Systolic CHT-RLS

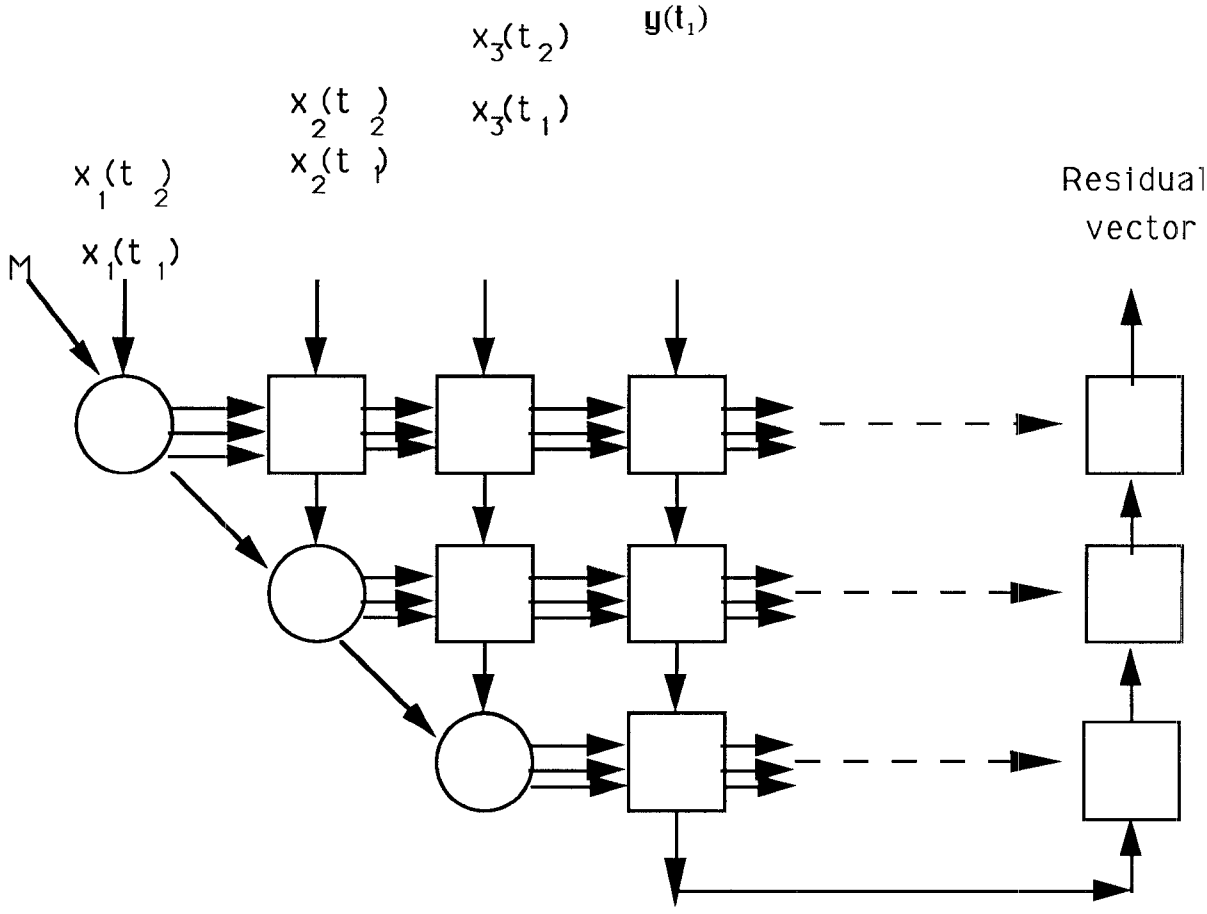


Figure 5.7: Systolic Architecture for CHT-RLS with Backward Propagation array

Initialization :	*Recursive* :
$r \leftarrow x(t_1)$	$e^{j\theta_r} = \frac{r}{\|r\|}$
$e^{j\theta_r} = \frac{r}{\|r\|}$	$s \leftarrow 0$
$s \leftarrow 0$	$for \;\; i = 0, M_{in}$
$for \;\; i = 1, M_{in}$	$w(t_i) \leftarrow r$
$w(t_i) \leftarrow r$	$s \leftarrow s + r^* r$
$s \leftarrow s + r^* r$	$r \leftarrow x(t_{i+1})$
$r \leftarrow x(t_{i+1})$	end
end	$\sigma \leftarrow \sqrt{s}$
$\sigma \leftarrow \sqrt{s}$	$r \leftarrow -e^{j\theta_r}\sigma$
$r \leftarrow -e^{j\theta_r}\sigma$	$\lambda \leftarrow \sigma(\sigma + \beta\|r\|)$
$\lambda \leftarrow \sigma(\sigma + \|r\|)$	$w(t_{M_{in}+1}) \leftarrow r$
$w(t_{M_{in}+1}) \leftarrow r$	$M_{out} \leftarrow M_{in}$
$M_{out} \leftarrow M_{in} - 1$	$H_{M\times M} = I_{M\times M} - \lambda \underline{x}_i \underline{x}_i^H$
	$\gamma_{out} = \gamma_{in} H_{M\times M}$

Table 5.3: The Algorithm for the Boundary Cell of Systolic CHT-RLS

Initialization :	*Recursive* :
$x(t_{M_{in}+1}) \leftarrow -x(t_1)$	$x(t_{M_{in}+1}) \leftarrow -r$
$s \leftarrow 0$	$s \leftarrow 0$
$for \;\; i = 1, M_{in}+1$	$for \;\; i = 0, M_{in}+1$
$s \leftarrow s + w^*(t_i)x(t_i)$	$s \leftarrow s + w^*(t_i)r$
end	$r \leftarrow x(t_{i+1})$
$\alpha \leftarrow \frac{s}{\lambda}$	end
$r \leftarrow -x(t_{M_{in}+1}) - \alpha w(t_1) + \alpha w(t_{M_{in}+1})$	$\alpha \leftarrow \frac{s}{\lambda}$
$for \;\; i = 2, M$	$r \leftarrow -\beta x(t_{M_{in}+1}) - \alpha w(t_0) + \alpha w(t_{M_{in}+1})$
$y(t_i) \leftarrow x(t_i) - \alpha w(t_i)$	$for \;\; i = 1, M$
end	$y(t_i) \leftarrow x(t_i) - \alpha w(t_i)$
	end

Table 5.4: The Algorithm for the Internal Cell of Systolic CHT-RLS

$$x_{out} = \gamma x_{out}$$

Table 5.5: The Algorithm for the Final Cell of Systolic CHT-RLS

Compared to the conventional adaptive array system, the QR-based systolic array system not only improves the numerical accuracy but also increases the computation speed to update the output signal instantly. The research in designing QR-based systolic arrays for sidelobe cancellation and adaptive antennas is quit intense. However, most of the work are based on Givens rotations and modified Gram-Schmidt. Only until recently, Householder-based method has just been considered [59]. In this paper, a CHT-RLS systolic array is designed for the application to sidelobe cancellation. The sidelobe cancellation technique is employed to suppress the sidelobe interference and noise by subtracting the estimate from the radar main channel output. It is easy to see from Figure 3.4 that the output at *ith* snapshot can be expressed as

$$e(i) = \sum_{l=1}^{N} x_l(i) w_l - y(i), \tag{5.57}$$

and the matrix form expression which is the same as 5.21 is given by

$$\underline{e}(n) = X(n)\underline{w}(n) - \underline{y}(n). \tag{5.58}$$

Therefore, the CHT-RLS systolic algorithm and architecture described in Section 5.3 can be directly applied to sidelobe cancellation and adaptive antennas to achieve high throughput rate and high speed requirements of real-time signal processing.

5.5 Conclusions

In order to achieve computational efficiency with robust numerical stability, the Householder transformation has been shown to be one of the best orthogonal factorizations. It is also known that the Householder transformation outperforms the Givens rotation in numerical stability under finite-precision implementation, and that it requires fewer arithmetic operations than the modified Gram-Schmidt does. As a result, the QR decomposition using the Householder transformation is very promising for VLSI implementation and real-time high throughput modern signal processing.

In this chapter, the recursive complex Householder algorithm with a fast initialization which can be programmed for both initialization and recursive computation is presented. Then a complex Householder transformation based recursive least squares algorithm with a systolic array processor is also proposed. Compared to the recursive block Householder algorithm described in [57,59], the complex Householder algorithm saves $(M-1) \times N$ computation time for the initialization to form the upper triangular matrix. In our proposed systolic architecture, the number of data snapshots needed for the initialization and recursive updating is controlled by a control code and it can be changed freely by sending another code. Our CHT-RLS systolic architecture is a generalization of McWhirter's QRD-RLS systolic array since it is well-known that the Givens rotation is a special case of the Householder transformation [55].

The algorithm and architecture described in this paper are very promising for the real-time array processing applications and VLSI hardware implementation.

Chapter 6

Conclusions and Future Research

In this dissertation several systolic algorithms are developed and then several different systolic array architectures are proposed for real-time high throughput adaptive array processing applications and VLSI hardware implementations. In Chapter 2 the design techniques and basic background for the field of algorithmic engineering are developed to serve as the basic elements of more sophisticated and higher performance array processing systems. In Chapter 3, the only two known systolic array processors for parallel weight extraction are developed for the recursive least squares (RLS) and constrained recursive least squares (CRLS) array systems [4]. In Chapter 4 a systolic array processor for MVDR beamforming is compared to the McWhirter's MVDR systolic

array. We also point out an error McWhirter and Shepherd made in their architecture [14]. In Chapter 5 the pipelined data-parallel algorithm for the Householder orthogonalization technique is described and mapped into a systolic array for generating the upper triangular matrix. Furthermore, a complex Householder-based recursive least squares (CHT-RLS) algorithm with systolic array architecture is presented. As described in [3], it is well-known that the Householder method outperforms the Givens method in numerical stability under finite-precision implementation, and that it requires fewer arithmetic operations than the modified Gram-Schmidt method. As a result, the QR decomposition using the Householder technique is very promising for VLSI implementation and real-time high throughput modern signal processing.

The proposed adaptive array systems in this dissertation have the following advantageous features:

- Since it is well-known that the QR based RLS and CRLS techniques have robust numerical properties, reduce the rounding error caused by finite word length effect, and avoid the use of sample matrix inversion. Our systolic array architecture based on QR decomposition is very promising for hardware implementation.

- In Chapters 2 and 3 there is no bottleneck in the proposed fast Givens based systolic array architectures since the fast Givens method is square root free.

- The proposed systolic array architectures in this dissertation are fully pipelined since the backward substitution processor in each proposed architecture is avoided.

- The proposed systolic array architectures are designed to be single fully pipelined open-loop systems without any feedback arrangement.

- The proposed systolic array architectures function recursively to update the output based on each input data snapshot.

- The proposed systolic architectures involve repetition of a simple processor element making it relatively easy to design a VLSI chip.

- The proposed systolic array architectures only involving regular communication among the nearest neighbor processor elements without feedback arrangement are very promising for the better use of available chip area, the reduction of propagation delays and the simpler VLSI circuit design.

- Since the proposed systolic array architectures are highly pipelined, they achieve the highest possible throughput rate.

Therefore, our proposed systolic array architectures are very promising for real-time modern signal processing applications and VLSI hardware implementation.

Future research will focus on (1) remedying the drawbacks of systolic arrays and (2) applying the systolic QR decomposition to the singular value decomposition and the symmetric eigendecomposition for superresolution spectral analysis and direction finding [29]. A list of the suggested research areas easily evolving from this dissertation includes:

- **Wavefront Array Processors.** Since the systolic arrays require global synchronization, there may be a problem with clock skew for a large arrays. The concept of wavefront array processors is the solution to this problem. Instead of using global synchronization, an asynchronous wavefront technique is used for wavefront array processors [45,46,47]. Systolic arrays studied in this dissertation can be easily extended into wavefront arrays.

- **Fault Tolerance.** For large arrays of processors on VLSI chips, the systolic arrays may have a serious reliability problem due to manufacturing defects, device failures, etc. Fault tolerance techniques for systolic arrays is the solution for this problem [26].

- **Wafer Scale Integration.** It is necessary to connect several chips on printed circuit boards to yield a large array due to integrated circuit fabrication limitations. The solution is a wafer scale integration technique. The advantage of this technique is that it may lead to higher

speeds of operation and less power consumption. Therefore, the wafer scale integration deserves study [25].

- **Singular Value Decomposition by Systolic Arrays.** In the applications of superresolution spectral analysis and direction finding, the singular value decomposition is one of the most powerful tools in numerical algebra. For real-time signal processing applications, it is important to study the parallel/pipelined algorithm and architecture for singular value decomposition [29]. It is not difficult to apply the systolic Householder method described in this dissertation to systolic singular value decomposition.

Bibliography

[1] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*. New York: Wiley, 1980.

[2] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[3] C.F. T. Tang, K.J. R. Liu, and S. A. Tretter, "On Systolic Arrays for Recursive Complex Householder Transformations with Applications to Array Processing," to appear, *Proc. of Int. Conf. on ASSP*, May, 1991.

[4] C.F. T. Tang, K.J. R. Liu, and S. A. Tretter, "A VLSI Algorithm and Architecture of CRLS Adaptive Beamforming," in press, *Proc. of the Conf. on Information Sciences and Systems*, March, 1991.

[5] B. L. Drake, J. M. Speiser, and J. J. Symanski, "SLAPP: A Systolic Linear Algebra Parallel Processor," *IEEE on Computer*, July, 1987, pp. 45-49.

[6] W. M. Gentleman and H. T. Kung, "Matrix Triangularization by Systolic Array," *Proc. SPIE*, Real-Time Signal Processing IV, 1981, **298**, pp. 19-26.

[7] H. T. Kung, "Why systolic architectures?" *Computer*, **15**, 37, 1982, pp. 37-45.

[8] M. Moonen and J. Vandewalle, "Recursive Least Squares with Stabilized Inverse Factorization," *Signal Processing*, Vol. 21, 1990, **1**, pp. 1-15.

[9] J. E. Hudson and T. J. Shepherd, "Parallel Weight Extraction by a Systolic Least squares Algorithm," *Proc. SPIE*, Advanced Algorithms and Architectures for Signal Processing IV, 1989, **1152**, pp. 68-77.

[10] T. J. Shepherd, J. G. McWhirter and J. E. Hudson, "Parallel Weight Extraction from a Systolic Adaptive Beamforming," *Mathematics in Signal Processing II,* 1990.

[11] J. G. McWhirter, "Algorithmic Engineering - an Emerging Discipline," *Proc. SPIE*, Advanced Algorithms and Architectures for Signal Processing IV, 1989, **1152**, pp. 2-15.

[12] R. Schreiber, "Implementation of Adaptive Array Algorithms," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-34, NO. 5, Oct. 1986, pp. 1038-1045.

[13] R. Schreiber and P. J. Kuekes, "Systolic Linear Algebra Machines in Digital Signal Processing," in *VLSI and Modern Signal Processing*, 1985, pp. 389-405.

[14] J. G. McWhirter and T. J. Shepherd, "Systolic Array Processor for MVDR Beamforming," *IEE proceedings*, Vol 136, No. 2, April 1989, pp. 75-80.

[15] B. Yang and J.F. Bome, "Systolic Implementation of A General Adaptive Array Processing Algorithm," *Proc. IEEE ICASSP*, April 1988, pp. 2785-2788.

[16] N. L. Owsley, "Systolic Array Adaptive Beamforming," *NUSC* report 7971, 1987.

[17] N. L. Owsley, "Systolic Array Adaptive Beamforming" in Haykin ed., *Selected Topics in Signal Processing*, Pentice-Hall, 1989.

[18] W. Givens, "Computation of Plane Unitary Rotations Transforming a General Matrix to Triangular Form," *J. Soc. Ind. Appl. Math.*, **6**, pp.26-50, 1958.

[19] W. M. Gentleman, "Least Squares Computations by Givens Transformation without Square Roots," *J.Inst. Maths Applics*, **12**, pp329-336, 1973.

[20] S. Hammarling, "A Note on Modifications to the Givens Plane Rotation," *J. Inst. Maths Applics*, **13**, pp 215-218, 1974.

[21] P. Comon and Y. Robert, "A Systolic Array for Computing BA^{-1}," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-35, No. 5 June. 1987, pp. 717-723.

[22] K.J. R. Liu and K. Yao, "Multi-Phase Systolic Algorithms for Spectral Decomposition," Accepted and to Appear in *IEEE Trans. Acoust. Speech, Signal Processing.*

[23] J. G. McWhirter and T. J. Shepherd, "A Systolic Array for Linearly Constrained Least-Squares Problems," *Proc. SPIE*, **696**, Advanced Algorithms and Architectures for signal Processing, 1986, pp. 80-86.

[24] J. G. McWhirter, "Recursive Least-Squares Minimization Using a Systolic Array," *Proc. SPIE*, **431**, Real-Time Signal Processing **VI**, 2983, 1983, pp. 105-112.

[25] C. M. Rader, D. B. Glasco, D. L. Allen, and C. E. Woodward, "MUSE - a Systolic Array for Adaptive Nulling with 64 Degrees of Freedom, Using Givens Transformations and Wafer Scale Integration," *Technical Report 886*, Lincoln Laboratory, MIT,1990.

[26] J. A. Abraham, P. Banerjee, C. Y. Chen, W. K. Fuchs, and S. Y. Kuo,

"Fault Tolerance Techniques for Systolic Arrays," *IEEE on Computer*, 1987, pp. 65-75.

[27] M. G. Bellanger, *Adaptive Digital Filter and Signal Analysis,* Marcel Dekker, Inc., New York and Basel, 1987.

[28] A. W. Bojanczyk and F. T. Luk, "A Novel MVDR Beamforming Algorithm," *Proc. SPIE*, Advanced Algorithms and Architectures for Signal Processing II, 1987, **826**, pp. 12-16. .

[29] R. Schreiber, "Bidiagonalization and Symmetric Tridiagonalization by Systolic Arrays," *Journal of VLSI Signal Processing*, **1**, 1990, pp. 279-285.

[30] S. Haykin,ed., *Array Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

[31] J. M. Cioffi, "The Fast Adaptive ROTOR's RLS Algorithm," *IEEE Trans. on ASSP*, Vol 38, No. 4, 1990, pp. 631-653.

[32] R. T. Compton Jr., *Adaptive Antennas: Concepts and Performance*, Prentice Hall, 1988.

[33] J. Gotze, B. Bruckmeier, and U. Schwiegelshohn, "VLSI-Suited Solution of Linear Systems," *IEEE ISCAS*, 1989, pp. 187-190.

[34] J. Gotze and U. Schwiegelshohn, "An Orthogonal Method for Solving Systems of Linear Equations Without Square Roots and with Few Divisions," *Proc. IEEE ICASSP*, 1989, pp. 1298-1301.

[35] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters*, Kluwer Academic Publishers, 1984.

[36] J. Cioffi, "The Fast Householder Filters RLS Adaptive Filter." *Proc. ICASSP 1990*, pp. 1619-1621.

[37] L. Johnson, "A Computational Array for The QR method," *1982 Conference on Advanced Research in VLSI*, MIT, pp. 123-129.

[38] G. Bienvenue and H.F. Mermoz, "New Principle of Array Processing in Underwater Passive Listening," in *VLSI and Modern Signal Processing*, S. Y. Whitehouse, and T. Kailath, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[39] K.M. Buckley, "Spatial/Spectral Filtering with Linearly Constrained Minimum Variance Beamformers," *IEEE Trans. Acoust. Speech, Signal Processing,* vol.**ASSP-35**, NO.3, pp.249-266, Mar. 1987.

[40] A. B. Fortes and B. W. Wah, "Systolic Arrays–From Concept to Implementation", *IEEE ON COMPUTER*, 1987.

[41] O. L. Frost III, "An Algorithm for Linearly Constrained Adaptive Array

Processing," *Proceedings of The IEEE*, vol. **60**,NO.8, pp.926-935, Aug. 1972.

[42] W, Givens, "Numerical Computation of the Characteristic Values of a Real Symmetric Matrix." Report **ORNL-1574**, Oak Ridge National Laboratory.

[43] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore and London, 1989

[44] S. Y. Kung, H. J. Whitehouse and T. Kailath (Eds.) *VLSI and Modern Signal Processing.* Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

[45] S. Y. Kung, *VLSI Array Processors.* Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[46] S. Y. Kung, "VLSI Array Processors," *IEEE ASSP Magazine*, 1985, pp. 4-22.

[47] S. Y. Kung, K. S. Arun, R. J. Gal-Ezer, and D. V. B. Rao, "Wavefront Array Processor: Language, Architecture, and Applications," *IEEE Trans. on Computer,* vol. c-31, No. 11, 1988, pp. 1054-1066.

[48] C. Mead and L. Conway, *Introduction to VLSI System* Addison-Wesley, Reading, Mass. 1980.

[49] S. K. Rao and T. Kailath, "VLSI Arrays for Digital Signal Processing. Part I: A Model Identification Approach to Digital Filter Realization. *IEEE Trans. Circuits Syst.*, **CAD-32**, 1105, 1985.

[50] R. Schreiber and P. J. Kuekes, in *VLSI and Modern Signal Processing.* Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

[51] L.H. Sibul and A.L. Fogelsanger, "Application of Coordinate Rotation Algorithm to Singular Value Decomposition," *PROC. of 1984 IEEE International Symposium on Circuits and Systems,* 1984.

[52] L.H. Sibul, "Application of Singular Value Decomposition to Adaptive Beamforming," *Proc. of ICASSP 84*, March 1984.

[53] B. D. Van Veen and K. M. Buckley, "Beamforming: a Versatile Approach to Spatial Filtering," *IEEE ASSP MAGAZINE*, pp. 4-24, Apr 1988.

[54] B.D. Van Veen and R.A. Roberts, "Systolic Arrays for Partially Adaptive Beamforming," *21-stAsilomar Conference on Signals, Systems, and Computers,* pp.584-588, Nov. 1987.

[55] Gene H. Golub and Charles F. Van Loan, *Matrix Computation.* The Johns Hopkins University Press 1983.

[56] Charles L. Lawson and Richard J. Hanson, *Solving Least Squares Problems.* Prentice-Hall Inc. New Jersey, 1974.

[57] K. J. R. Liu, S. F. Heieh, and K. Yao, "Recursive LS Filtering using Block Householder Transformations." *Proc. IEEE ICASSP*, 1990, pp. 1631-1634.

[58] K. J. R. Liu, S. F. Heieh, and K. Yao, "Two Level Pipelined Implementation of Systolic Block Householder Transformations with Application to RLS Algorithm." *Proc. Int'l Conf. on Application-Specific Array Processors*, pp. 748-769, Princeton, Sep. 1990.

[59] K.J. R. Liu, S. F. Heieh, and K. Yao, "Systolic Block Householder Transformation for RLS Algorithm with Two-lwvel Pipelined Implementation," to appear in *IEEE Trans. on Signal Processing*, April 1992.

[60] N. K. Tsao, "A Note on Implementing the Householder Transformation," *SIAM J. Numer. Anal.,* Vol. 12, No. 1, pp. 53-58, 1975.

[61] James W. Longley, *Least Squares Computations Using Orthogonalization Method.* Marcel Dekker Inc. 1984.

[62] Charles M. Rader and Allan O. Steinhardt, "Hyperbolic Householder Transformations." *IEEE Trans. on ASSP*, vol ASSP-34, No. 6 DEC. 1986, pp. 1589-1602.

[63] R. Schreiber and W. P. Tang, "On Systolic Arrays for Updating the Cholesky Factorization," Swedish Roy. Inst. Technol., Dep. Numeric.

Anal. Comput. Sci., Stockholm, Sweden, Tech. Rep. **TRITA-NA-8313**, 1983.

[64] Allan O. Steinhardt, "Householder Transformations in Signal Processing." *IEEE ASSP Magazine*, July 1988, pp. 4-12.

[65] C.R.Ward, P.J. Hargrave, and J.G. McWhirter, "A Novel Algorithm and Architecture for Adaptive Digital Beamforming," *IEEE Trans. on AP,* **AP-34**, pp. 338-346, Mar. 1986.

[66] C.R.Ward, A. J. Robson, P.J. Hargrave, and J.G. McWhirter, "Application of a Systolic Array to Adaptive Beamforming," *IEE Proceedings*, Vol. 131, Pt. F, **6**, Oct. 1984, pp. 638-645.

[67] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford 1965.

[68] Stanley M. Yuen, "Algorithmic, Architectural, and Beam Pattern Issues of Sidelobe Cancellation," *IEEE on AES*, VOL.25, **NO.4**, July 1989, pp. 459-472.

[69] F. Ling, D. Manolakis, and J. G. Proakis, "A Recursive Modified Gram-Schmidt Algorithm for Least-Squares Estimation," *IEEE Trans. on ASSP*, VOL.34, **NO.4**, 1986, pp. 829-836.

[70] S. Z. Kalson and K. Yao, "A Class of Least-Squares Filtering and Identification Algorithms with Systolic Array Architectures," *IEEE Trans. on*

Information Theory, Vol 37, Jan. 1991, pp. 43-52.

[71] J. G. Nash and S. Hansen, "Modified Faddeeva Algorithm for Concurrent Execution of Linear Algebraic Operations," *IEEE Trans. on Computer*, Vol 37, Feb. 1988, pp. 129-137.

[72] P. Strobach, *Linear Prediction Theory: A Mathematical Basis for Adaptive Systems*, Springer-Verlag 1990.

[73] J. R. Treichler, Jr. C. R. Johnson, and M. G. Larimore, *Theory and Design of Adaptive Filters*, John Wiley & Sons, New York, 1987.