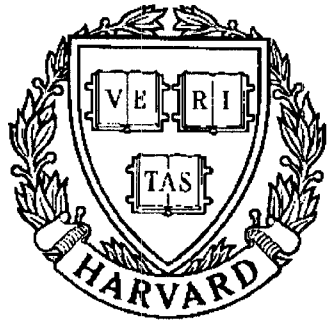
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



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

A Fully Parallel and Pipelined Systolic Array for MVDR Beamforming

by C.F.T. Tang and K.J.R. Liu

TR 91-89

A Fully Parallel and Pipelined Systolic Array for MVDR Beamforming

C.F. T. Tang and K.J. R. Liu
Electrical Engineering Department
and Systems Research Center
University of Maryland
College Park, MD 20742

Abstract

A fully-pipelined systolic array for computing the minimum variance distortionless response (MVDR) was first proposed by McWhirter and Shepherd. The fundamental concept is to fit the MVDR beamforming to the non-contrainted recursive least-squares (RLS) minimization. Until now, their systolic array processor is well-recognized as the most efficient design for MVDR beamforming. In this paper, we first point out the mistake by relating the MVDR beamforming and RLS minimization and then propose a new algorithm for the MVDR beamforming. Moreover, a fully parallel and pipelined systolic array for the newly proposed algorithm is presented and the square-root free implementation is also considered.

1 Introduction

Research in implementing the minimum variance distortionless response (MVDR) algorithm by systolic array processors to compute the optimal residual has been very active in the last few years [1, 2, 3, 4, 6, 7]. 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 onto a systolic array processor are (1) numerical stability, (2) computational efficiency, and (3) single fully-pipelined structure. In a recent paper a numerically stable and computationally efficient algorithm for MVDR beamforming was introduced by Schreiber [1]. 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, nevertheless, it is difficult to implement the whole algorithm onto a single fully-pipelined structure as pointed out in [3, 4]. Lately, McWhirter and Shepherd [4] proposed a single fully-pipelined systolic array processor (SAP) for minimum variance distortionless response (MVDR) beamforming by implementing Schreiber's algorithm without the need of an extra back substitution processor for computing the residual [1, 4]. Up to now, McWhirter and Shepherds' systolic array processor is well-recognized as the most efficient design for MVDR beamforming. Their SAP for MVDR beamforming is designed by using the recursive least squares (RLS) algorithm proposed in McWhirter's earlier paper [5]. According to McWhirter and Shepherds' paper, by comparing the MVDR algorithm to the RLS algorithm, it seems easy to obtain the residual of the MVDR beamforming from that of the RLS algorithm by forcing the desired input data to be zero. Unfortunately, the residual of the RLS algorithm described in [5] is derived by employing the lower part of a unitary matrix, S, while the residual of MVDR is obtained by using the upper part of a unitary matrix, P, where the up-

per part and lower part of the unitary matrix $Q(n)$ is defined as $Q(n) = \begin{bmatrix} P(n) \\ S(n) \end{bmatrix}$. In this paper, we first point out that the algorithm for McWhirter and Shepherds' MVDR beamforming to compute the residual by forcing the desired data to zero is not correct and then propose a new and correct systolic algorithm and systolic array processor to compute the optimal residuals for MVDR beamforming [6, 7].

This paper is organized as follows. In the second section McWhirter's RLS and MVDR algorithms are summarized and the problem of using the residual algorithm of RLS for MVDR is pointed out. In the third section, the newly proposed MVDR algorithm with single fully-pipelined systolic array processors is presented in detail. In the fourth section, a single fully-pipelined MVDR systolic array processor is illustrated and the operations for its processor elements are also described. In the fifth section, the square-root free MVDR beamforming is considered. Finally, in the last section, we conclude our results.

2 Comparison Between McWhirter's RLS and MVDR Algorithms

In this section, the problem of using the RLS for MVDR algorithm is addressed and McWhirter's RLS and MVDR algorithms are summarized. We claim that MVDR can not use RLS simply by driving the desired data $y(t_n)$ to zero. The correct MVDR algorithm should 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 necessary to be examined again in detail. The comparison of the McWhirter's two systolic array processors is very important to have a clear picture of

the approach described in this paper. Therefore, we start with a detailed description of McWhirter's work and then present a correct solution for computing the residual for MVDR beamforming. In the next two subsections McWhirter's algorithm for RLS minimization is summarized and the MVDR algorithm for computing the residual by driving the desired data to be zero is described [5].

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

$$\underline{e}(n) = X(n)\underline{w}(n) - \underline{y}(n), \tag{1}$$

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

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

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

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

$\underline{w}(n)$, 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,

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

with $B(n)$, an exponentially forgetting matrix which emphasizes the statistical weight on the new data and gradually scales down the old data in the least squares compu-

tation, given by

$$B(n) = \begin{bmatrix} \beta^{n-1} & 0 & 0 & 0 & 0 \\ 0 & \beta^{n-2} & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where $0 < \beta < 1$.

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

$$Q(n)X(n) = \begin{bmatrix} R(n) \\ 0 \end{bmatrix}. \tag{2}$$

The $n \times n$ 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,

$$Q(n) = \begin{bmatrix} P(n) \\ S(n) \end{bmatrix}. \tag{3}$$

Also, the QR decomposition can be applied to the desired data vector $\underline{y}$. Then, we have

$$Q(n)\underline{y}(n) = \begin{bmatrix} P(n)\underline{y}(n) \\ S(n)\underline{y}(n) \end{bmatrix} = \begin{bmatrix} \underline{u}(n) \\ \underline{v}(n) \end{bmatrix}. \tag{4}$$

The L_2 norm of the residual is

$$\begin{aligned} ||\underline{e}(n)||_2 &= ||X(n)\underline{w}(n) - \underline{y}(n)||_2 \\ &= ||Q^H(\begin{bmatrix} R(n) \\ 0 \end{bmatrix}\underline{w}(n) - \begin{bmatrix} \underline{u}(n) \\ \underline{v}(n) \end{bmatrix})||_2 \\ &= ||\begin{bmatrix} R(n)\underline{w}(n) - \underline{u}(n) \\ -\underline{v}(n) \end{bmatrix}||_2. \end{aligned} \tag{5}$$

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

$$\underline{w}_{LS}(n) = R^{-1}(n)\underline{u}(n), \tag{6}$$

and hence the minimized $||\underline{e}||_2$ is

$$||\underline{e}(n)||_2 = ||\underline{v}(n)||_2, \tag{7}$$

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

It was shown in [5] that the unitary matrix $Q(n)$ can be factorized as follows:

$$Q(n) = \hat{Q}(n)\overline{Q}(n-1), \tag{8}$$

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

$$\overline{Q}(n-1) = \begin{bmatrix} P(n-1) & 0 \\ S(n-1) & 0 \\ 0 & 1 \end{bmatrix}, \tag{9}$$

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

$$\hat{Q}(n) = \begin{bmatrix} A(n) & 0 & \underline{a}(n) \\ 0 & I & 0 \\ \underline{b}^T(n) & 0 & \gamma(n) \end{bmatrix}. \tag{10}$$

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

$$P(n) = \begin{bmatrix} A(n)P(n-1) & \underline{a}(n) \end{bmatrix}, \tag{11}$$

and

$$S(n) = \begin{bmatrix} S(n-1) & 0 \\ \underline{b}^T(n)P(n-1) & \gamma(n) \end{bmatrix}. \tag{12}$$

By substituting (6) into (1), the residual vector is found to be

$$\underline{e}_{LS}(n) = X(n)R^{-1}(n)\underline{u}(n) - \underline{y}(n). \tag{13}$$

The derivation of the current residual at the n^{th}-snapshot for recursive least squares described by McWhirter is given as follows. According to (13), we have

$$\underline{e}_{LS}(n) = Q^H(n) \begin{bmatrix} R(n) \\ 0 \end{bmatrix} R^{-1}(n)\underline{u}(n) - Q^H(n) \begin{bmatrix} \underline{u}(n) \\ \underline{v}(n) \end{bmatrix}. \tag{14}$$

Therefore, the residual vector has the form

$$\begin{aligned} \underline{e}_{LS}(n) &= -Q^H(n) \begin{bmatrix} 0 \\ \underline{v}(n) \end{bmatrix} \\ &= -S^T(n)\underline{v}(n), \end{aligned} \tag{15}$$

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

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

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

$$\underline{v}(n) = \begin{bmatrix} \beta\underline{v}(n-1) \\ \alpha(n) \end{bmatrix}, \tag{16}$$

with $\alpha(n)$ given by

$$\alpha(n) = \beta\underline{b}^T(n)\underline{u}(n-1) + \gamma(n)y(t_n). \tag{17}$$

The current residual $e_{LS}(t_n)$, i.e., the last element of $\underline{e}_{LS}(n)$, is then given by

$$e_{LS}(t_n) = -\gamma(n)\alpha(n) = -\gamma(n)(\beta\underline{b}^T(n)\underline{u}(n-1) + \gamma(n)y(t_n)). \tag{18}$$

where $\gamma(n)$ and $\underline{b}^T(n)$ are given in the Appendix. (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 subsection, McWhirter's systolic array for the recursive least-squares problem illustrated in [5] is shown in Figure 1.

2.2 McWhirter's MVDR Beamforming

Let the $n \times 1$ residual vector for the MVDR problem be

$$\underline{e}(n) = X(n)\underline{w}(n), \tag{19}$$

where $X(n)$ and $\underline{w}(n)$ are the same as that of in the least squares problem. Mathematically, the MVDR problem is to find a $\underline{w}(n)$ that solve the following optimization problem,

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

where $\underline{c}$ is the steering vector and K is the number of constrained directions. Using the method of Lagrange multipliers, the optimal weight vector $\underline{w}^i_{opt}(n)$ is

$$\underline{w}^i_{opt}(n) = \frac{r^i M^{-1}(n)\underline{c}^i}{\underline{c}^{iH} M^{-1}(n)\underline{c}^i} \quad for \ \ i = 1, \cdots, K, \tag{20}$$

where

$$M(n) = X^H(n)X(n). \tag{21}$$

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

$$Q(n)X(n) = \begin{bmatrix} R(n) \\ 0 \end{bmatrix}. \tag{22}$$

Substituting (22) into (21), we have

$$M(n) = R^H(n)R(n). \tag{23}$$

Therefore, the optimal weight vector is

$$\underline{w}_{opt}^{i} = \frac{r^{i} R^{-1}(n) R^{-H}(n) \underline{c}^{i}}{\underline{c}^{iH} R^{-1}(n) R^{-H}(n) \underline{c}^{i}} \quad for \;\; i = 1, \cdots, K. \tag{24}$$

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

$$\underline{z}^{i}(n) = R^{-H}(n) \underline{c}^{i}. \tag{25}$$

The upper triangular matrix can be updated by using the QR decomposition given by

$$\widehat{Q}(n) \begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^{T}(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix}, \tag{26}$$

where $\widehat{Q}(n)$ has the form as given in (10).

From (25), we have

$$\begin{aligned} \underline{c}^{i} &= R^{H}(n) \underline{z}^{i}(n) \\ &= R^{H}(n-1) \underline{z}^{i}(n-1) \\ &= \begin{bmatrix} \beta R^{H}(n-1) & 0 & \underline{x}^{*}(t_n) \end{bmatrix} \widehat{Q}^{H}(n) \widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta} \underline{z}^{i}(n-1) \\ \# \\ 0 \end{bmatrix}, \end{aligned} \tag{27}$$

and from (26), the same QR decomposition can be applied to update the parameter $\underline{z}^{i}(n-1)$, so that

$$\widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta} \underline{z}^{i}(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{z}^{i}(n) \\ \# \\ \# \end{bmatrix}, \tag{28}$$

where # denotes an arbitrary vector of no interest in mathematical and physical concept. On substituting (25) and (24) into (19), the residual vector for MVDR beamforming is

$$\underline{e}_{MVDR}(n) = \frac{r^{i}}{||\underline{z}^{i}(n)||^{2}} X(n) R^{-1}(n) \underline{z}^{i}(n). \tag{29}$$

For convenience, we define a vector $\underline{\widehat{e}}_{MVDR}(n)$ as

$$\underline{\widehat{e}}_{MVDR}(n) = X(n)R^{-1}(n)\underline{z}^i(n). \tag{30}$$

Obviously, the current residual can be obtained by evaluating

$$\widehat{e}_{MVDR}(t_n) = \underline{x}^T(t_n)R^{-1}(n)\underline{z}^i(n). \tag{31}$$

By changing the data matrix $X(n)$ and the desired data vector $\underline{y}(n)$ into a data vector $\underline{x}^T(t_n)$ and a desired data $y(t_n)$ at the n^{th} snapshot in (13), the current residual of the RLS algorithm can be written as

$$e_{RLS}(t_n) = \underline{x}^T(t_n)R^{-1}(n)\underline{u}(n) - y(t_n). \tag{32}$$

Comparing to (31) and (32), it seems that by forcing the desired data $y(t_n)$ to zero, $\underline{u}(n)$ to $\underline{z}^i(n)$, and β to $\frac{1}{\beta}$, the current residual for MVDR beamforming can be obtained by using that of the RLS algorithm [4]. Therefore, according to (18), the residual for MVDR beamforming seems to be

$$\begin{aligned} \widehat{e}_{MVDR}(t_n) &= -\gamma(n)\alpha(n) \\ &= -\frac{1}{\beta}\gamma(n)\underline{b}^T\underline{z}^i(n-1). \end{aligned} \tag{33}$$

Comparing the two residual equations for RLS and MVDR shown in [4], 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 [4], the systolic array processor for MVDR beamforming has the structure given by Figure 2.

The question now arises "Can MVDR beamforming use the RLS algorithm by driving the desired data to zero to compute the residual?" The answer is NO because given the zero desired data, the algorithm to compute the current residual for MVDR at the n^{th}-snapshot is not the same as McWhirter's RLS algorithm. It is also known

that when the desired data set to be zero continually, the residuals will be zero eventually. By a simple derivation in the next section, we point out that the algorithm for McWhirter's MVDR beamforming, using (33) to compute the residual by forcing the desired data $y(t_n)$ in the (32) to be zero, is not correct. The new and correct algorithm to compute the current residual for MVDR beamforming will then be given.

3 A Novel MVDR algorithm

Recall that the residual vector for MVDR beamforming is

$$\underline{e}_{MVDR}(n) = \frac{r^i}{||\underline{z}^i(n)||^2} X(n)R^{-1}(n)\underline{z}^i(n). \tag{34}$$

For convenience, define a vector $\widehat{\underline{e}}_{MVDR}(n)$ as in (30)

$$\widehat{\underline{e}}_{MVDR}(n) = X(n)R^{-1}(n)\underline{z}^i(n). \tag{35}$$

By substituting (22) into (35), we have

$$\begin{aligned} \widehat{\underline{e}}_{MVDR}(n) &= Q^H(n) \begin{bmatrix} R(n) \\ 0 \end{bmatrix} R^{-1}(n)\underline{z}^i(n) \\ &= Q^H(n) \begin{bmatrix} \underline{z}^i(n) \\ 0 \end{bmatrix} = P^H(n)\underline{z}^i(n) \\ &= \begin{bmatrix} P^H(n-1)A^H(n)\underline{z}^i(n) \\ \underline{a}^H(n)\underline{z}^i(n) \end{bmatrix}, \end{aligned} \tag{36}$$

where $Q^H(n)$ is

$$Q^H(n) = \begin{bmatrix} P^H(n) & S^H(n) \end{bmatrix},$$

and from (11) $P^H(n)$ is

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

Obviously, here $\underline{\widehat{e}}_{MVDR}(n)$ depends only on $P(n)$, instead of $S(n)$ in the RLS case. On substitution (10) into (28), $\underline{z}^i(n)$ is given by

$$\begin{bmatrix} \underline{z}^i(n) \\ \# \\ \# \end{bmatrix} = \widehat{Q}(n) \begin{bmatrix} \frac{1}{\beta}\underline{z}^i(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} A(n)\frac{1}{\beta}\underline{z}^i(n-1) \\ \# \\ \# \end{bmatrix}, \tag{37}$$

Therefore, the current residual associated with (35) for MVDR beamforming has the form

$$\begin{aligned} \widehat{e}_{MVDR}(t_n) &= \underline{a}^H(n)\underline{z}^i(n) = \frac{1}{\beta}\underline{a}^H(n)A(n)\underline{z}^i(n-1) \\ &= \frac{1}{\beta}(s_1(n)c_1(n)z_1(n-1) + c_1(n)s_2(n)(-s_1(n)s_2^*(n)z_1(n-1) \\ &\quad +c_2(n)z_2(n-1)) + \cdots\cdots\cdots + (c_1(n)\cdots c_{n-1}(n)s_n(n)) \\ &\quad (-s_1(n)c_2\cdots c_{n-1}(n)s_n^*(n)z_1(n-1) - s_2(n)c_3(n)\cdots c_{n-1}(n) \\ &\quad s_n^*(n)z_2(n-1) - \cdots\cdots + c_n(n)z_n(n-1))), \end{aligned} \tag{38}$$

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

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

and $A(n)$ is also given in the Appendix.

Recall that (33) given by McWhirter and Shepherd can be evaluated as

$$\begin{aligned} \widehat{e}_{MVDR}(t_n) &= -\frac{1}{\beta}\gamma(n)\underline{b}^T\underline{z}^i(n-1) \\ &= -\frac{1}{\beta}((c_1(n)c_2(n)\cdots c_n(n))(-s_1(n)c_2(n)\cdots c_n(n)z_1(n-1) - \cdots\cdots \\ &\quad -s_{n-1}(n)c_n(n)z_{n-1}(n-1) - s_n(n)z_n(n-1))). \end{aligned} \tag{39}$$

Comparing (38) and (39), it is readily seen that both equations are absolutely different. The current residual of the MVDR beamforming is given by

$$e^i_{MVDR}(t_n) = \frac{r^i}{||\underline{z}^i(n)||^2}\underline{a}^H(n)\underline{z}^i(n). \tag{40}$$

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 is used for the period $0 \leq n \leq N$. Initialization algorithm must be used first before the recursive algorithm is employed. The initialization and the recursive updating are described in the following subsection.

3.1 Initialization Procedure

The initialization 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 $\underline{z}^i(N)$. The QR decomposition applied to the first N data snapshots $X(N)$ to obtain the initial upper triangular matrix has the form given by

$$Q(N)X(N) = R(N). \tag{41}$$

A vector $\underline{z}^i(N)$ is obtained by the forward substitution, so that

$$\underline{z}^i(N) = R^{-H}(N)\underline{c}^i. \tag{42}$$

3.2 Recursive Updating

The recursive algorithm is applied for time $n > N$ to update and compute the residuals. It is well known [4] that the QR decomposition of the observed data $X(n)$ can be implemented recursively on a triangular systolic array. Recall from (26), by applying the QR decomposition, the upper triangular matrix can be updated, so that

$$\widehat{Q}(n)\begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix}, \tag{43}$$

where $\widehat{Q}(n)$ is given in (10).

From (27), the parameter $\underline{z}^i(n-1)$ can be updated by using the same QR decomposition, so that

$$\widehat{Q}(n)\begin{bmatrix} \frac{1}{\beta}\underline{z}^i(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{z}^i(n) \\ \# \\ \# \end{bmatrix}, \tag{44}$$

Therefore, recall from (40) the current residual is given by

$$\begin{aligned} e^i_{MVDR}(t_n) &= \frac{r^i}{||\underline{z}^i(n)||^2}\underline{a}^H(n)\underline{z}^i(n) \\ &= \frac{r^i}{||\underline{z}^i(n)||^2}\sum_{j=1}^{n} s_j c_{j-1}\cdots c_1 z_j. \end{aligned} \tag{45}$$

The newly developed MVDR algorithm is summarized in Table 1. The residuals obtained by this algorithm is defined only during the recursive updating for $n \geq N$ when the observed data matrix is of full rank. First, the upper triangular matrix and a parameter vector are initialized by setting to be zero, i.e., $R(0) = 0$ and $\underline{z}(0) = \underline{0}$. In the initialization, mode 1 and mode 2 are required while only mode 1 is needed in recursive updating. In the initialization for $0 \leq n \leq N$, when the observed input data matrix $X(N)$ is available, where N is the number of sensors, the initial upper triangular $R(N)$ is generated by the QR decomposition called the mode 1 operation and the initial parameter vector $\underline{z}^i(N)$ is computed by parallel multiplication instead of the forward substitution and this is called the mode 2 operation. Finally, for the recursive updating during the time period $n > N$, the upper triangular matrix $R(n-1)$ and the parameter matrix $\underline{z}^i(n-1)$ can be updated and at the same time, the current residual of MVDR is obtained by the multiplication and accumulation operation using the updated data $R(n-1)$ and $\underline{z}^i(n-1)$ under the mode 1 operation only.

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

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

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

Table 1: Summary of Parallel/Pipelined QRD-MVDR Algorithm

4 A Novel MVDR Systolic Array Processor

In Figure 3, a single fully-pipelined MVDR systolic array processor for N sensors and K constraints to receive the observed input vector and steering vector, and then to instantaneously generate the updated residuals in parallel is given for the case of $N = 4$ and $K = 2$. The system needs two procedures which are the initialization and the recursive updating. The initialization is further divided into two parts. First, under the mode 1 operation, the 4×4 observed input data X, and the 4×2 zero matrix are fed into the MVDR systolic array to compute the 4×4 upper triangular matrix R stored in $PE1$ and $PE2$ processor elements. Second, under the mode 2 operation, the 2×4 steering vector and 2×2 identity matrix are sent into the processor to generate the 4×2 matrix stored in $PE3$ processor elements. The upper triangular processor carries out the QR decomposition under the mode 1 operation and under the mode 2 operation, it performs parallel multiplication and accumulation operation instead of the forward substitution to generate a 4×2 matrix when a 2×2 identity matrix is received, and the $PE3$s function as loading elements when 1's are received. Finally, during recursive updating, the residuals are obtained in parallel under the mode 1 operation. When a 1×4 observed input data vector and 1×2 zero vector are fed into the processor array the updated residuals are obtained instantaneously at the bottom of the array. Note that $PE4$ processor elements perform the normalization.

The four processor elements are depicted in Figure 4. Compared to Figure 2, McWhirter and Shepherds' MVDR systolic array, the boundary cells send parameters into the internal cells as well as the next boundary cell. However, the boundary cells in Figure 4 only send parameters to internal cells. The mode 1 operation for the MVDR systolic array for each processor element is described in Table 2, and the mode 2 operation is presented in Table 3. The mode 1 operation is used to carry out the QR decomposition and parallel multiplication in both initialization and

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

Table 2: The Operation for Mode 1

recursive updating. Under mode 1 operation, the processor element 1 generates the rotation coefficients c and s when zeroing out the observed input data. The processor element 2 performs the rotation of the received input data according to the rotation coefficients. The processor element 3 not only performs the loading operation but also carries out the parallel multiplication and accumulation operation to compute the non-normalized residuals. The processor element 4 performs the normalization to generate the residuals. The mode 2 operation is employed to carry out the parallel multiplication and accumulation operation to generate the 4×2 matrix and to store them into $PE3$ processor elements. In the mode 2 operation, the processor element 1 is used to generate parameters while the processor element 2 is employed to carry out the parallel multiplication and accumulation operation. The processor element 3 is simply used to store the matrix. It is illustrated in Figure 3 how the two different modes described in Tables 2 and 3 are performed in the initialization. The recursive updating for computing the residuals for $n > N$ only requires mode 1 operation [6, 8].

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

Table 3: The Operation for Mode 2

5 Fast Givens Based-MVDR Beamforming

The upper triangular matrix $R(n)$ can be rewritten as

$$R(n) = D^{\frac{1}{2}}(n)\overline{R}(n) = Q(n)X(n) \tag{46}$$

where $D(n)$ is a N by N diagonal matrix with the 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 with the form

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

We can also factor the diagonal matrix $D^{\frac{1}{2}}$ out of the orthogonalized desired vector $\underline{z}^i$ as

$$\underline{z}^i(n) = D^{\frac{1}{2}}\overline{\underline{z}^i}(n). \tag{47}$$

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

$$Q(N)X(N) = D^{\frac{1}{2}}(N)R(N), \tag{48}$$

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

$$\overline{\underline{z}^i}(N) = \overline{R}^{-H}(N)\underline{c}^i. \tag{49}$$

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

$$\begin{aligned}\widehat{Q}(n)\begin{bmatrix} \beta D^{\frac{1}{2}}(n-1)\overline{R}(n-1) & \vdots & \frac{1}{\beta}D^{\frac{1}{2}}(n-1)\overline{\underline{z}^i}(n-1) \\ 0 & \vdots & \# \\ \underline{x}^T(t_n) & \vdots & 0 \end{bmatrix} \\ = \begin{bmatrix} D^{\frac{1}{2}}(n)\overline{R}(n) & \vdots & D^{\frac{1}{2}}(n)\overline{\underline{z}^i}(n) \\ 0 & \vdots & \# \\ 0 & \vdots & 0 \end{bmatrix}.\end{aligned} \tag{50}$$

Finally,

$$e^i_{MVDR}(t_n) = \frac{r^i}{\overline{\underline{z}^i}^H(n)D^{-1}(n)\overline{\underline{z}^i}(n)}\underline{a}^H(n)D^{-1}(n)\overline{\underline{z}^i}(n) \quad for \;\; i = 1, \cdots, K. \tag{51}$$

The algorithm of the fast Givens MVDR with modifications from those of the Givens MVDR is described in Tables 4. Similar to the Givens MVDR, we start with setting the initial condition to be zero, i.e., $D^{\frac{1}{2}}(0)\overline{R}(0) = 0$ and $\overline{\underline{z}^i}(0) = 0$. In the initialization for $0 \leq n \leq N$, the diagonal matrix $D^{\frac{1}{2}}(N)$ and the upper triangular matrix $\overline{R}(N)$ are obtained under the mode 1 operation, the QR decomposition, and the parameter vector $\overline{\underline{z}^i}(N)$ is then computed under the mode 2 operation, multiplication and accumulation operation [8]. In the recursive updating for $n > N$, the diagonal matrix, the upper triangular matrix, and the parameter vector are updated and the current residual can also be computed simultaneously under the mode 1 operation. The fast Givens MVDR systolic array processors is illustrated in Figure 5.

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

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

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) $\underline{\overline{z}}^i(N) = \overline{R}^{-H}(N)\underline{c}^i$ (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) $$\hat{Q}(n)\begin{bmatrix} \frac{1}{\beta} D^{\frac{1}{2}}(n-1)\underline{\overline{z}}^i(n-1) \\ \# \\ 0 \end{bmatrix} = \begin{bmatrix} D^{\frac{1}{2}}(n)\underline{\overline{z}}^i(n) \\ \# \\ \# \end{bmatrix}$$

   (c) $e^i_{MVDR}(t_n) = \frac{r^i}{\underline{z}^{iH}(n)D^{-1}(n)\underline{z}^i(n)} \sum_{j=1}^{n} s_j c_{j-1} \cdots c_1 \frac{z_j}{d(j)}$

Table 4: Summary of Parallel/Pipelined Fast Givens-MVDR Algorithm

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

Table 5: The Fast Operation 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 6: The Fast Operation of Mode 2

The four processor elements of the fast Givens MVDR systolic array are depicted in Figure 6. In the fast Givens MVDR SAP, the boundary cell not only sends parameters to the internal cell but also to the next boundary cell for avoiding square root computation. Their operations for mode 1, QR decomposition and parallel multiplication, and mode 2, the multiplication and accumulation operation, are stated in Table 5 and 6. In Table 5, the mode 1 operation performs the fast Givens rotations and parallel multiplication. In addition, under the mode 1 operation, the processor element 1 is used to generate parameters without computing the square root, the processor element 2 is operated to transform the input data by those parameters, and the processor element 3 is employed not only to load the data but also to carry out the accumulation and addition operation for obtaining the non-normalized residuals. The processor element 4 performs as the normalization operation to compute the residuals. In Table 6, the mode 2 operation functions as the parallel multiplication and accumulation operation without the need for forward substitution. The functions of the four processor elements in the fast MVDR systolic array processor are performed in the same way as in the MVDR systolic array processor described in Section 4 [6, 8].

6 Conclusion

In this paper, we first point out the mistake in [4] and argue the problem by directly applying the RLS algorithm for MVDR beamforming. Conceptually, we see that if the desired data is set to zero in the RLS problem, it is no longer a LS minimization problem since the solution is unique, that is, the optimal weight vector is uniquely defined and the residuals should be eventually become zeros. Then a new approach to design a single fully-pipelined systolic array processor for MVDR beamforming by implementing Schreiber's algorithm in [1] to compute the residual is proposed. The square-root free implementation is also considered for the newly proposed architecture.

Appendix

To update the upper triangular matrix $R(n-1)$, we have

$$\widehat{Q}(n)\begin{bmatrix} \beta R(n-1) \\ 0 \\ \underline{x}^T(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ 0 \end{bmatrix}, \tag{52}$$

In order to obtain the elements of $\widehat{Q}(n)$ which updates the upper triangular matrix based on a new row vector, we start from a 2×2 to a 4×4 upper triangular matrices and then generalize from them for a $n\times n$ case. For 2×2 upper triangular matrix and a new row vector, $\widehat{Q}(2)$ can be obtained by the following steps and also be factored into Q_2Q_1. First, Q_1 can be carried out to zero x_1, so that

$$\begin{aligned} Q_1\begin{bmatrix} \beta r_{11} & \beta r_{12} \\ 0 & \beta r_{22} \\ x_1 & x_2 \end{bmatrix} &= \begin{bmatrix} c_1 & 0 & s_1^* \\ 0 & 1 & 0 \\ -s_1 & 0 & c_1 \end{bmatrix}\begin{bmatrix} \beta r_{11} & \beta r_{12} \\ 0 & \beta r_{22} \\ x_1 & x_2 \end{bmatrix} \\ &= \begin{bmatrix} r_{11}^{'} & r_{12}^{'} \\ 0 & \beta r_{22} \\ 0 & x_2^{'} \end{bmatrix}, \end{aligned} \tag{53}$$

where

$$r_{11}^{'} \leftarrow \sqrt{(\beta r_{11})^2 + |x_1|^2},$$

$$c_1 \leftarrow \frac{\beta r_{11}}{r_{11}^{'}},$$

$$s_1 \leftarrow \frac{x_1}{r_{11}^{'}},$$

$$r_{11}^{'} \leftarrow c_1\beta r_{11} + s_1^* x_1,$$

and

$$r_{12}^{'} \leftarrow c_1\beta r_{12} + s_1^* x_2.$$

Then, Q_2 is applied to zero out x_2, we have

$$Q_2 \begin{bmatrix} r_{11}^{'} & r_{12}^{'} \\ 0 & \beta r_{22} \\ 0 & x_2^{'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2^* \\ 0 & -s_2 & c_2 \end{bmatrix} \begin{bmatrix} r_{11}^{'} & r_{12}^{'} \\ 0 & \beta r_{22} \\ 0 & x_2^{'} \end{bmatrix}$$
$$= \begin{bmatrix} r_{11}^{'} & r_{12}^{'} \\ 0 & r_{22}^{'} \\ 0 & 0 \end{bmatrix}, \tag{54}$$

where $r_{22}^{'} \leftarrow \sqrt{(\beta r_{22})^2 + |x_2|^2}$.

Therefore, $\widehat{Q}(2)$ is obtained by the multiplication of Q_2 and Q_1 given by

$$\widehat{Q}(2) = \begin{bmatrix} c_1 & 0 & s_1^* \\ 0 & 1 & 0 \\ -s_1 & 0 & c_1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2^* \\ 0 & -s_2 & c_2 \end{bmatrix}$$
$$= \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & c_1 & 0 & \vdots & s_1^* & \vdots \\ \vdots & -s_1 s_2^* & c_2 & \vdots & c_1 s_2^* & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & -s_1 c_2 & -s_2 & \vdots & c_1 c_2 & c_1 c_2 \end{bmatrix}. \tag{55}$$

Similarly, $\widehat{Q}(3)$ is given by

$$\widehat{Q}(3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_3 & s_3^* \\ 0 & 0 & -s_3 & c_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & s_2^* \\ 0 & 0 & 1 & 0 \\ 0 & -s_2 & 0 & c_2 \end{bmatrix} \begin{bmatrix} c_1 & 0 & 0 & s_1^* \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_1 & 0 & 0 & c_1 \end{bmatrix}$$

$$= \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & c_1 & 0 & 0 & \vdots & s_1^* & \vdots \\ \vdots & -s_1 s_2^* & c_2 & 0 & \vdots & c_1 s_2^* & \vdots \\ \vdots & -s_1 c_2 s_3^* & -s_2 s_3^* & c_3 & \vdots & c_1 c_2 s_3^* & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & -s_1 c_2 c_3 & -s_2 c_3 & -s_3 & \vdots & c_1 c_2 c_3 & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \tag{56}$$

and, $\widehat{Q}(4)$ is

$$\widehat{Q}(4) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & c_4 & s_4^* \\ 0 & 0 & 0 & -s_4 & c_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & c_3 & 0 & s_3^* \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -s_3 & 0 & c_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 & s_2^* \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -s_2 & 0 & 0 & c_2 \end{bmatrix} \begin{bmatrix} c_1 & 0 & 0 & 0 & s_1^* \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -s_1 & 0 & 0 & 0 & c_1 \end{bmatrix}$$

$$= \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & c_1 & 0 & 0 & 0 & \vdots & s_1^* & \vdots \\ \vdots & -s_1 s_2^* & c_2 & 0 & 0 & \vdots & c_1 s_2^* & \vdots \\ \vdots & -s_1 c_2 s_3^* & -s_2 s_3^* & c_3 & 0 & \vdots & c_1 c_2 s_3^* & \vdots \\ \vdots & -s_1 c_2 c_3 s_4^* & -s_2 c_3 s_4^* & -s_3 s_4^* & c_4 & \vdots & c_1 c_2 c_3 s_4^* & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & -s_1 c_2 c_3 c_4 & -s_2 c_3 c_4 & -s_3 c_4 & -s_4 & \vdots & c_1 c_2 c_3 c_4 & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \end{bmatrix}. \tag{57}$$

Finally, by induction $\widehat{Q}(n)$ is given by

$$\widehat{Q}(n) = \begin{bmatrix} A(n) & \underline{a}(n) \\ \underline{b}^T(n) & \gamma(n) \end{bmatrix}, \tag{58}$$

where

$$A(n) = \begin{bmatrix} c_1 & 0 & 0 & \cdots & 0 \\ -s_1 s_2^* & c_2 & 0 & \cdots & 0 \\ -s_1 c_2 s_3^* & -s_2 s_3^* & c_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -s_1 c_2 \cdots c_{n-1} s_n^* & -s_2 c_3 \cdots c_{n-1} s_n^* & -s_3 c_4 \cdots c_{n-1} s_n^* & \cdots & c_n \end{bmatrix}, \tag{59}$$

$$\underline{a}^T(n) = \begin{bmatrix} s_1^* & c_1 s_2^* & c_1 c_2 s_3^* & \cdots & c_1 \cdots c_{n-1} s_n^* \end{bmatrix}, \tag{60}$$

$$\underline{b}^T(n) = \begin{bmatrix} -s_1 c_2 \cdots c_n & -s_2 c_3 \cdots c_n & -s_3 c_4 \cdots c_n & \cdots & -s_n \end{bmatrix}, \tag{61}$$

and

$$\gamma(n) = c_1 c_2 \cdots c_{n-1} c_n. \tag{62}$$

References

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

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

[3] 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. .

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

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

[6] C.F. T. Tang, *Adaptive Array Systems Using QR-Based RLS and CRLS Techniques with Systolic Array Architectures*, Ph.D Thesis Report, Ph.D.91.5, Systems Research Center, University of Maryland, College Park, May, 1991.

[7] C.F. T. Tang and K.J. R. Liu, "A Novel Systolic Array for MVDR Beamforming," Sumitted to *IEEE Int. Conf. on ASSP 1992.*

[8] C.F. T. Tang, K.J. R. Liu, and S. A. Tretter, "Parallel and Fully-Pipelined Instantaneous Optimal Weight Extraction for Adaptive Beamformer Using Systolic Arrays," Submitted to *IEEE Transactions on Aerospace and Electronic Systems.*

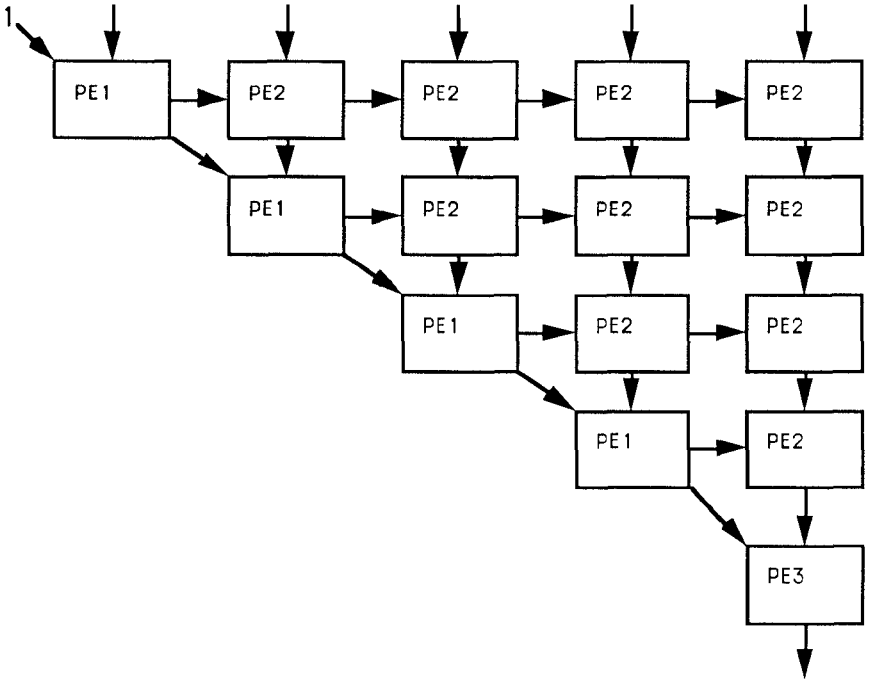


Figure 1: McWhirter's RLS Systolic Array Processor

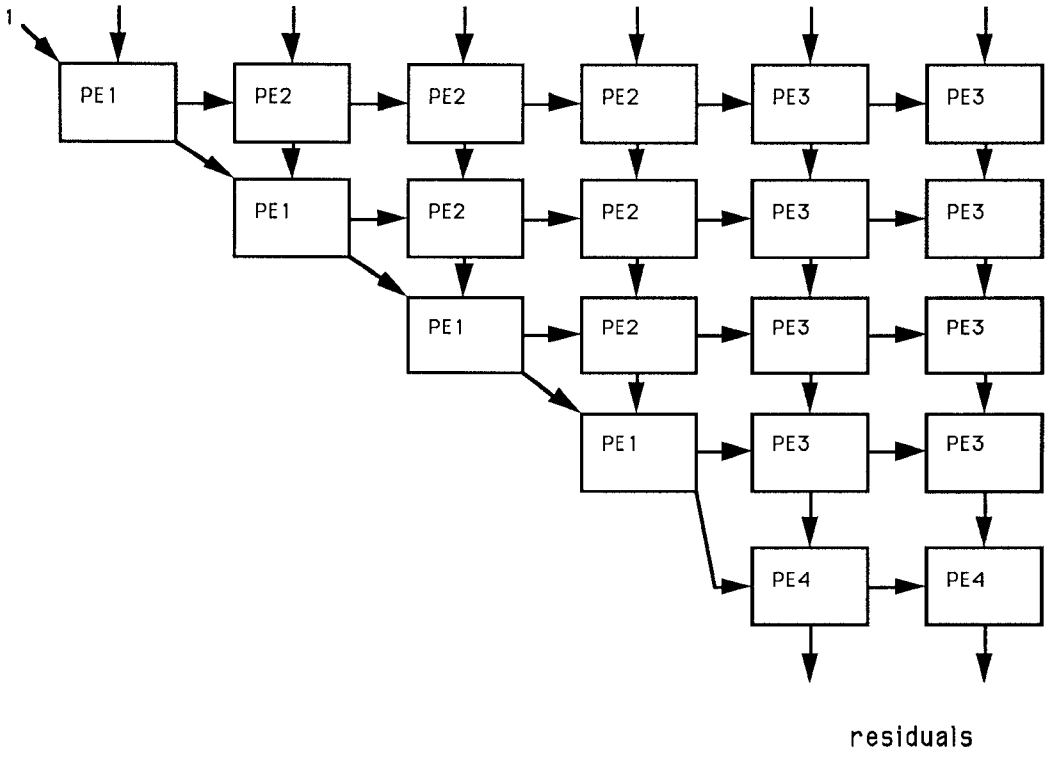


Figure 2: McWhirter's MVDR Systolic Array Processors

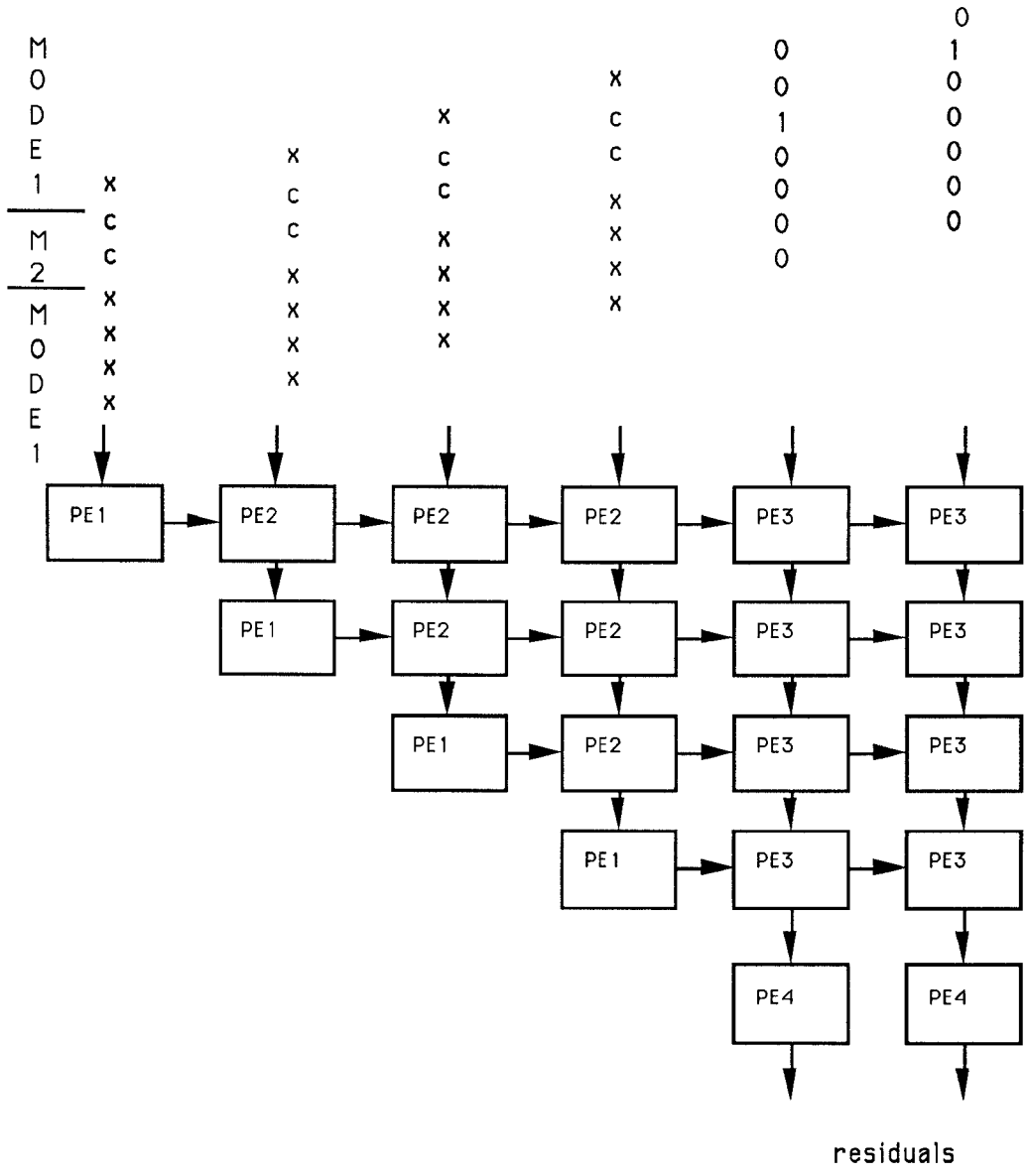


Figure 3: MVDR Systolic Array Processor

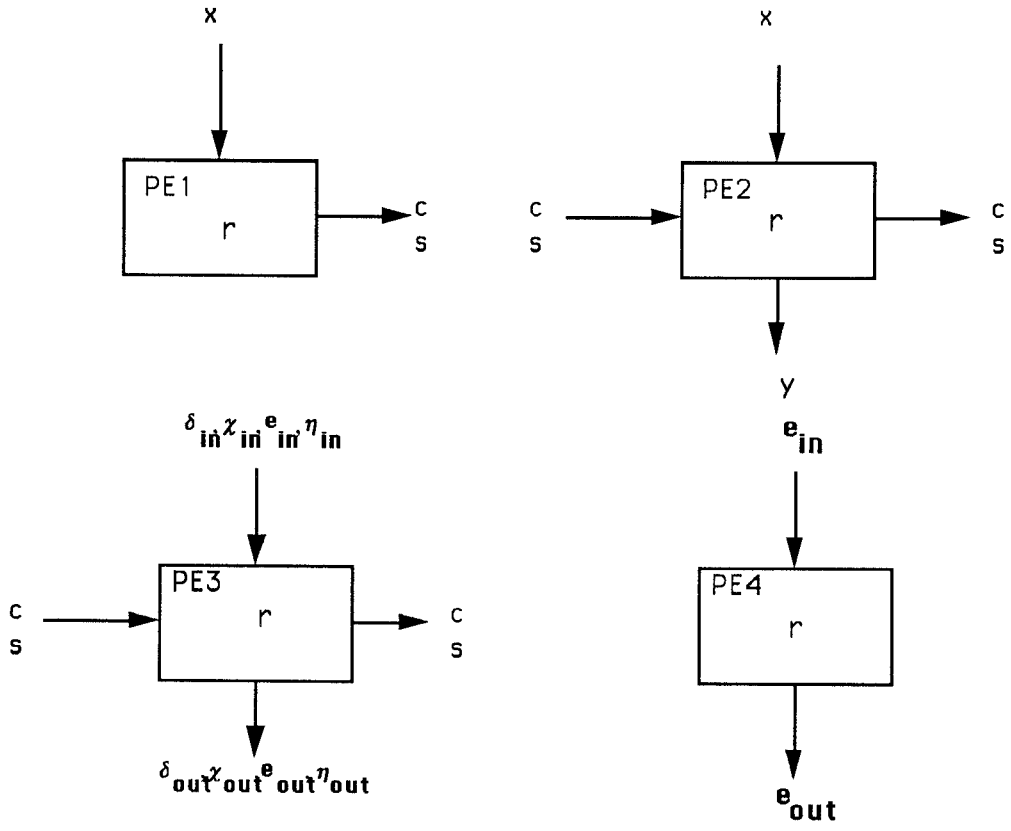


Figure 4: Processor Elements of MVDR Systolic Array

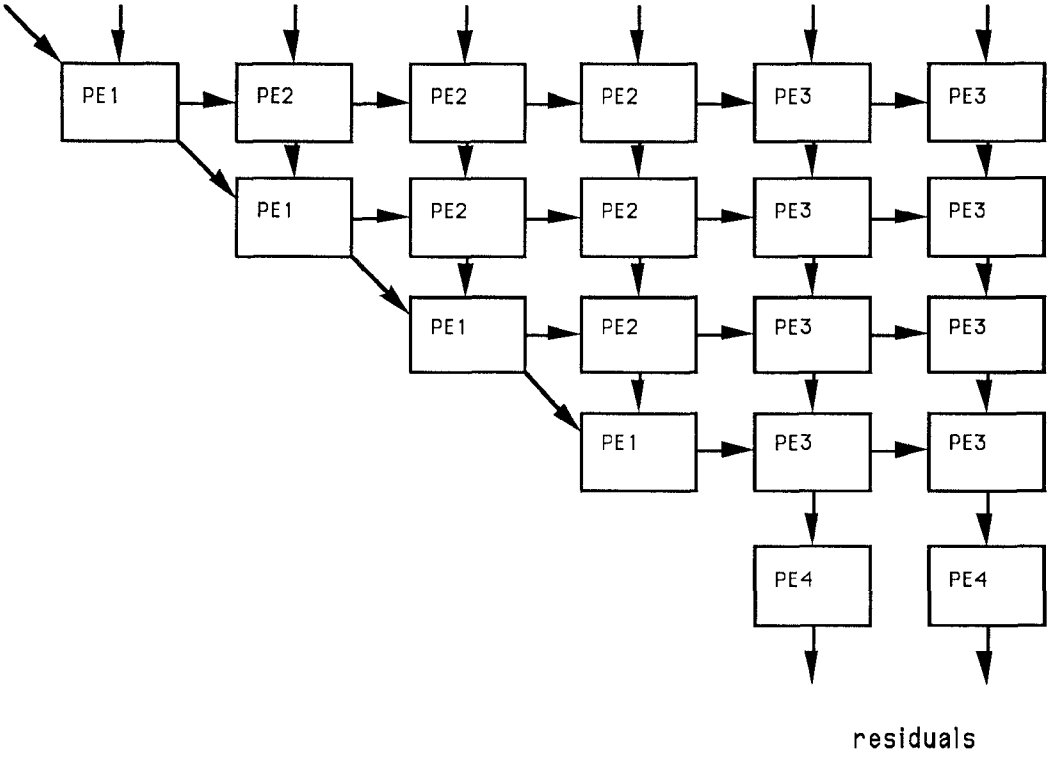


Figure 5: Fast MVDR Systolic Array Processor

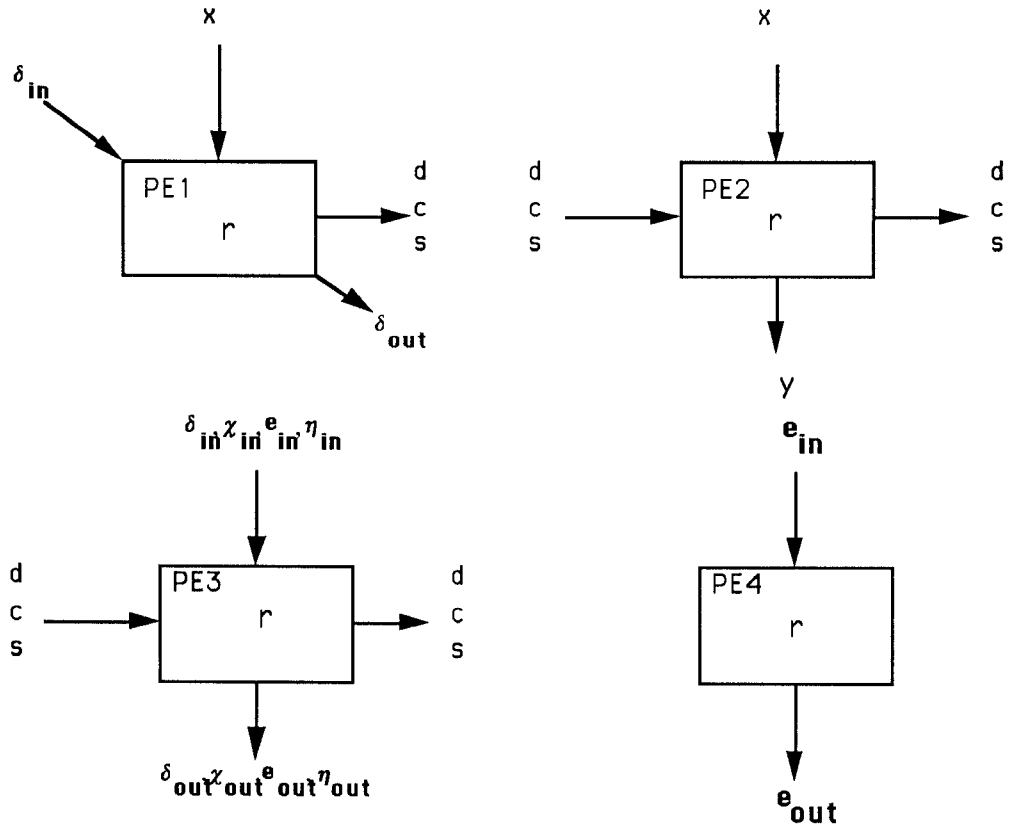


Figure 6: Processor Element of Fast MVDR Systolic Array