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

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

by C.F.T. Tang, K.J.R. Liu, S.F. Hsieh, and K. Yao

TR 91-84

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

C.F. T. Tang, K.J. R. Liu, S.F. Hsieh[1], and K. Yao[2]

Systems Research Center

Electrical Engineering Department

University of Maryland

College Park, MD 20742

Abstract

The Householder transformation is considered to be desirable among various unitary transformations due to its superior computational efficiency and robust numerical stability. Specifically, the Householder transformation outperforms the Givens rotation and the modified Gram-Schmidt methods in numerical stability under finite-precision implementations, as well as requiring fewer arithmetical operations. Consequently, the QR decomposition based on the Householder transformation is promising for VLSI implementation and real-time high throughput modern signal processing. In this paper, a recursive complex Householder transformation with a fast initialization algorithm is proposed and its associated parallel/pipelined architecture is also considered. Then, a complex Householder transformation based recursive least-squares algorithm with a fast initialization is presented. Its associated systolic array processing architecture is also considered.

[1]Dept. of Communication, Engineering, Nat'l Chiao Tung University, Hsinchu, Taiwan 30039

[2]Electrical Engineering Department, UCLA, Los Angeles, CA 90024-1594

1 Introduction

In recent years, there has been considerable interests in mapping QR decomposition (QRD) algorithms onto systolic arrays. One motivation 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 as well as superior numerical accuracy. Therefore, the need of finite precision computations in VLSI systems has involved the use of numerical analysis techniques in modern signal processing.

The QRD 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, etc. One of the reasons is the robust numerical stability in these implementations since the rounding errors caused by finite word length effects will not be accumulated in using the QR-RLS approaches. Another reason is that QR-RLS algorithms can be mapped efficiently onto systolic arrays which are promising candidates for high speed real-time signal processing applications and VLSI implementations. Basically, there are three approaches to the QR-RLS problem: namely, Householder transformation, Givens rotation method, and the modified Gram-Schmidt method. The Givens rotation method can also be considered to be a special case of the Householder transformation [1].

Systolic array implementations for QR-RLS algorithms have been explored by numerous researchers. Gentleman and Kung [2] first introduced the linear least-squares (LS) problem based on the Givens rotation method with systolic array implementation to derive the optimal weights. Their method for the LS computation consists of two steps based on the orthogonal triangularization and the backward substitution. Step one was carried out on a triangular systolic array for the QR decomposition and step two was implemented on 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 on the systolic array implementation for the LS problem. McWhirter [3] then showed that the Givens rotations based RLS algorithm can be implemented efficiently on a single systolic array for directly computing the residual without the need of the backward substitution array. The resulting Givens based-RLS array is both fully parallel and pipelined. Other results on systolic Givens rotation [4, 5, 6] and systolic Cholesky decomposition [4, 7] are known. Independently, Ling et al. [8] and Kalson and Yao [9, 10] proposed recursive modified Gram-Schmidt (RMGS) algorithms for LS estimations as well as fully pipelined architectures for computing the residuals. Other systolic array implementations for QR-RLS algorithms have also been explored [11, 12, 13, 14, 15].

Until recently, most of the systolic array implementations for the QR-RLS algorithms have been based on the Givens rotation and the modified Gram-Schmidt methods. Recently, the general problem of designing algorithms based on the Householder transformation (HT) and its associated systolic architectures has been considered [17, 18, 19, 20, 21, 22]. In [17], Johnsson proposed a computational array for the complex householder transformation. Unfortunately, his computational array involves the globe communication instead of the neighbor communication as systolic array does. In [18, 19], the applications of Householder transformation to signal processing are introduced. Rader and Steinhardt pointed out in [18] that by applying the QR methods to sidelobe canceller, the complex HT not only has the lowest SNR loss under the finite wordlength effects but also is the least expensive computationally. It has been shown that the HT generally outperforms the Givens rotation and the modified Gram-Schmidt methods under finite precision computations [17, 18, 21, 22, 23]. It is also known that the Householder method requires less computation than the Givens and modified Gram-Schmidt methods. In [20], Cioffi introduced the fast Householder-RLS adaptive filter but it seems difficult to implement this algorithm in parallel form. Most recently, algorithms and architectures for the systolic block HT based RLS (SBHT-RLS) have been presented in [21, 22]. However, in many practical signal processing applications, we need to deal with complex-valued signals. Therefore, in this paper, we develop complex HT versions of the above HT systolic algorithms and architectures for real-time high throughput modern signal processing applications. Furthermore, as compared to the SBHT-RLS method, the systolic architecture for the complex HT based RLS (CHT-RLS) algorithm developed in this paper saves $(N-1)M$ units in computation time for obtaining 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 [21, 22]. In this 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 readily by sending another control code into the cells.

This paper is organized as follows. Section 2 consists of a complex Householder transformation algorithm and its associated systolic array processor [15, 16]. We begin with the development of a systolic complex Householder algorithm which is programmable to handle both the initialization and recursive computations. Then we introduce the systolic architectures for the parallel complex Householder algorithms. A two-level pipelined implementation of the complex Householder transformation is also considered. Section 3 consists of a CHT-RLS systolic algorithm and its systolic array implementation [16]. First, a systolic algorithm for CHT-RLS with fast initialization is described. Then, the systolic array processor of the CHT-RLS systolic algorithm is proposed. Section 4 consists of some discussions on the application of the CHT-RLS array for sidelobe cancellation. Finally, the Conclusion is given in Section 5.

2 Systolic Algorithms and Arrays for Complex Householder Transformation

In many signal processing applications, a numerically stable and efficient approach for performing the QRD is needed. There are many schemes to perform such a decomposition, e.g., Givens/modified Givens, Householder, and 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 be 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 only when there are more than N data snapshots.

2.1 Systolic Complex Householder Algorithm

The following Lemma shows how a Householder transformation can be applied to a column vector $\underline{x}$ to zero out all elements except the first one. First a column vector $\underline{u}$ is defined from a given column vector $\underline{x}$, then the complex Householder transformation B is computed from the defined column vector $\underline{u}$, and finally the elements below the first element of the given column vector $\underline{x}$ are zeroed out by applying the computed complex Householder transformation to it.

Lemma [1]

Suppose $\underline{x} = [x_1, \ldots, x_k]^T \in \mathbf{C}^k$ 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 $k \times k$ 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 the complex conjugate transpose.

2.1.1 Initialization

The factorization of a data matrix $X \in \mathbf{C}^{N\times N}$ can be achieved by a sequence of Householder transformations [21, 22] which produces a unitary $N \times N$ matrix Q and an upper triangular matrix R such that

$$X = Q^H R,$$

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

$$Q = Q_{N-1} \cdots Q_2 Q_1 \tag{1}$$

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

$$Q_i = \begin{bmatrix} I_{i-1 \times i-1} & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & B_i \end{bmatrix}, \qquad for\ i = 1, \cdots, N-1, \tag{2}$$

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

$$B_i = I - \frac{2}{\underline{u}_i^H \underline{u}_i} \underline{u}_i \underline{u}_i^H, \tag{3}$$

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

$$\underline{u}_i^T = \begin{bmatrix} x_i(t_i) + e^{j\theta_i(t_i)}||\underline{x}_i||_2 & x_i(t_{i+1}) & \cdots & x_i(t_N) \end{bmatrix} \tag{4}$$

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

$$(B_i \underline{x}_i)^T = \begin{bmatrix} r_{ii} & 0 & \cdots & 0 \end{bmatrix}, \qquad for \;\; i = 1, 2, \cdots, N-1, \tag{5}$$

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

Define a scalar parameter λ as

$$\lambda = \frac{2}{\underline{u}_i^H \underline{u}_i} = (||\underline{x}_i||_2 (||\underline{x}_i||_2 + |x_i(t_i)|))^{-1}. \tag{6}$$

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

$$B_i \underline{x}_k = \underline{x}_k - \lambda \underline{u}_i \underline{u}_i^H \underline{x}_k = \underline{x}_k - \alpha \underline{u}_i, \tag{7}$$

where α is a scalar parameter given by

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

and

$$B_i \underline{x}_k = \begin{bmatrix} x_k(t_i) - \alpha x_i(t_i) + \alpha r_{ii} \\ x_k(t_{i+1}) - \alpha x_i(t_{i+1}) \\ \vdots \\ x_k(t_N) - \alpha x_i(t_N) \end{bmatrix}, \qquad for \;\; k = i+1, \cdots, N. \tag{8}$$

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 $(N-1) \times (N-1)$ unitary matrix B_2 is applied to the $(N-1) \times (N-1)$ 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 an upper triangular form by applying $N-1$ unitary transformations to it. It is well known that the number of arithmetical operations gradually decreases in each of the subsequent Householder transformations.

2.1.2 Recursive Complex Householder Algorithm

The triangular matrix R can be updated by employing unitary Householder transformations P, so that

$$P\begin{bmatrix}\beta R\\ X\end{bmatrix}=\begin{bmatrix}R'\\ 0\end{bmatrix}, \tag{9}$$

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 represents a sequence of complex Householder transformations used to zero out the new complex $M\times N$ data matrix X, and β is the forgetting factor. The sequence of Householder transformations to update the triangular matrix R is given by

$$P=P_NP_{N-1}\cdots P_2P_1, \tag{10}$$

where an $(M+N)\times(M+N)$ complex Householder transformation P_i has the form

$$P_i=\begin{bmatrix}I_{i-1\times i-1} & \vdots & 0\\ \cdots & \cdots & \cdots\\ 0 & \vdots & B_i\end{bmatrix}, \qquad for \quad i=1,2,\cdots,N, \tag{11}$$

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

$$B_i=I-\frac{2}{\underline{u}_i^H\underline{u}_i}\underline{u}_i\underline{u}_i^H. \tag{12}$$

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

$$\underline{x}_i'^T=\begin{bmatrix}\beta r_{ii} & 0 & \cdots & 0 & x_i(t_i) & \cdots & x_i(t_M)\end{bmatrix}, \tag{13}$$

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

$$\underline{u}_i^T = \left[\begin{array}{ccccccc} \beta r_{ii} + e^{j\theta_{r_{ii}}} \|\underline{x}_i^{'}\|_2 & 0 & \cdots & 0 & x_i(t_i) & \cdots & x_i(t_M) \end{array} \right], \tag{14}$$

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

$$B_i = \left[\begin{array}{ccccc} H_{1\times 1} & \vdots & 0 & \vdots & H_{1\times M} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \vdots & I_{N-i\times N-i} & \vdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ H_{M\times 1} & \vdots & 0 & \vdots & H_{M\times M} \end{array} \right], \tag{15}$$

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

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

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

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

and

$$\lambda = \frac{2}{\underline{u}_i^H \underline{u}_i} = (\|\underline{x}_i^{'}\|_2(\|\underline{x}_i^{'}\|_2 + \beta|r_{ii}|))^{-1}. \tag{16}$$

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

$$(B_i \underline{x}_i^{'})^T = \left[\begin{array}{cccc} r_{ii}^{'} & 0 & \cdots & 0 \end{array} \right], \qquad for \quad i = 1, 2, \cdots, N, \tag{17}$$

where $r_{ii}^{'} = -e^{j\theta_{r_{ii}}} \|\underline{x}_i^{'}\|_2$. When the same Householder transformation is applied to the rest of the column vector $\underline{x}_k^{'}$ with

$$\underline{x}_k^{'T} = \left[\begin{array}{ccccccccc} \beta r_{ik} & \cdots & \beta r_{kk} & 0 & \cdots & 0 & x_k(t_i) & \cdots & x_k(t_M) \end{array} \right],$$

the new set of column vectors can by obtained by

$$B_i \underline{x}_k^{'} = \begin{bmatrix} \beta r_{ik} - \alpha r_{ii} + \alpha r_{ii}^{'} \\ \beta r_{i+1\ k} \\ \vdots \\ \beta r_{kk} \\ 0 \\ \vdots \\ 0 \\ x_k(t_i) - \alpha x_i(t_i) \\ \vdots \\ x_k(t_M) - \alpha x_i(t_M) \end{bmatrix}, \quad for \quad k = i+1, \cdots, N, \tag{18}$$

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

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.

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 1 shows the boundary cell and internal cell of the systolic recursive Householder transformations. The operation of the boundary cell is given in Table 1 and that of the internal cell is described in Table 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 3.

Figure 2 illustrates the operation carried out by a given pair of boundary and internal cells. The data flow from the boundary cell to internal cell is pipelined sample-by-sample. The boundary and internal cells operate 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 1, the newly updated data r which replaces the previous r is also sent to the internal cell. The data flow and computations 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 HT algorithm [24] is given in [22]. The two-level pipelined architectures for the conventional and modified Householder transformation have some similarities as well as differences. Compared to the two-level pipelined architecture shown in [22], the architecture based on conventional Householder transformation described in Figure 2 requires one more multiplication and addition computation than that of the modified Householder transformation. The similarity between the 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 3.

In adaptive antenna and radar applications, the period of updating the optimum weights is significantly larger than the actual computation time [26], 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 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 circuitry and programmable capabilities.

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

Table 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 2: The Algorithm for the Internal Cell of Complex Householder Transformation

3 Systolic CHT-RLS Algorithm and Architecture

In this section we consider the solution of the RLS problem based on the systolic 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 given by

$$X(1:k) = \begin{bmatrix} \underline{x}^T(t_1) \\ \underline{x}^T(t_2) \\ \vdots \\ \underline{x}^T(t_k) \end{bmatrix} \tag{19}$$

and

$$\underline{y}(1:k) = \begin{bmatrix} y(t_1) \\ y(t_2) \\ \vdots \\ y(t_k) \end{bmatrix}. \tag{20}$$

The LS residual vector is given by

$$\underline{e}(1:k) = X(1:k)\underline{w}(k) - \underline{y}(1:k), \tag{21}$$

where the weights for each of N sensors are

$$\underline{w}(k) = \begin{bmatrix} w_1(t_k) \\ w_2(t_k) \\ \vdots \\ w_N(t_k) \end{bmatrix}, \tag{22}$$

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

$$E(1:k) = ||X(1:k)\underline{w}(k) - \underline{y}(1:k)||_2. \tag{23}$$

The solution to this minimization problem is

$$\underline{w}_{opt}(k) = (X^H(1:k)X(1:k))^{-1}X^H(1:k)\underline{y}(1:k). \tag{24}$$

3.1 Systolic CHT-RLS Algorithm

The approach described in (24) is generally known as direct sample matrix inversion (SMI) method. It is well known that the classical SMI method, may sometimes lead to undesirable numerical characteristics due to ill-conditioned matrices. This means that an extremely high arithmetical precision is required when employing the SMI method. To alleviate such roundoff sensitivity caused by the SMI method, the QR decomposition deserves serious consideration. In this section, we introduce the CHT-LS algorithm to compute the residuals. In this algorithm, 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.

3.1.1 Initialization

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 [21, 22] and produces a unitary N by N matrix $Q(N)$ and an $N \times N$ upper triangular matrix $R(N)$ such that

$$X(1:N) = Q^H(N)R(N), \tag{25}$$

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 LS residual vector for the initialization procedure is

$$\underline{e}(1:N) = X(1:N)\underline{w}(N) - \underline{y}(1:N). \tag{26}$$

Therefore,

$$\begin{aligned} E(1:N) &= ||Q(N)X(1:N)\underline{w}(N) - Q(N)\underline{y}(1:N)||_2 \\ &= ||R(N)\underline{w}(N) - \underline{u}(N)||_2. \end{aligned} \tag{27}$$

Since $Q(N)$ is unitary, according to (27), the optimal weight vector is given by

$$R(N)\underline{w}_{opt}(N) - \underline{u}(N) = \underline{0} \tag{28}$$

3.1.2 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

$$X(1:k) = Q^H(k) \begin{bmatrix} R(k) \\ 0 \end{bmatrix}, \tag{29}$$

where $R(k)$ is an $N \times N$ upper triangular matrix. The LS residual vector is

$$\underline{e}(1:k) = X(1:k)\underline{w}(k) - \underline{y}(1:k). \tag{30}$$

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

$$\begin{aligned} E(1:k) &= ||Q(k)X(1:k)\underline{w}(k) - Q(k)\underline{y}(1:k)||_2 \\ &= || \begin{bmatrix} R(k) \\ 0 \end{bmatrix} \underline{w}(k) - \begin{bmatrix} \underline{u}(k) \\ \underline{v}(k) \end{bmatrix} ||_2, \end{aligned} \tag{31}$$

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, as

$$Q(k) = \begin{bmatrix} Q_1(k) \\ Q_2(k) \end{bmatrix}, \tag{32}$$

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

$$Q_1(k)\underline{y}(1:k) = \underline{u}(k), \tag{33}$$

$$Q_2(k)\underline{y}(1:k) = \underline{v}(k). \tag{34}$$

To minimize the quantity $E(1:k)$, the optimal weight vector obtained from (31) has the form as

$$R(k)\underline{w}_{opt}(k) = \underline{u}(k), \tag{35}$$

Then, substituting (29) and (35) into (30), the residual vector is given by

$$\hat{e}(1:k) = \begin{bmatrix} 0 \\ -Q_2 v(k) \end{bmatrix}. \tag{36}$$

Therefore, the residual vector for the LS problem is

$$\hat{e}((N+1):k) = -Q_2^H v(k). \tag{37}$$

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

$$X(1:(k+M)) = \begin{bmatrix} \beta X(1:k) \\ X((k+1):(k+M)) \end{bmatrix}, \tag{38}$$

and the desired data vector is given by

$$\underline{y}(1:(k+M)) = \begin{bmatrix} \underline{y}(1:k) \\ \underline{y}((k+1):(k+M)) \end{bmatrix}, \tag{39}$$

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

$$Q(k+M) = P(k+M)\overline{Q}(k) \tag{40}$$

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

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

The desired HT $P(k+M)$ described in [21, 22] for updating the upper triangular matrix has the form

$$P(k+M) = \begin{bmatrix} H_{11}(k+M) & 0 & H_{12}(k+M) \\ 0 & I_{(k-N)\times(k-N)} & 0 \\ H_{21}(k+M) & 0 & H_{22}(k+M) \end{bmatrix}, \tag{41}$$

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

$$\overline{Q}(k)X(1:(k+M)) = \begin{bmatrix} \beta R(k) \\ 0 \\ X((k+1):(k+M)) \end{bmatrix}. \tag{42}$$

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

$$\begin{aligned} Q(k+M)X(1:(k+M)) &= P(k+M)\begin{bmatrix} \beta R(k) \\ 0 \\ X((k+1):(k+M)) \end{bmatrix} \\ &= \begin{bmatrix} R(k+M) \\ 0 \\ 0 \end{bmatrix}, \end{aligned} \tag{43}$$

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 CHT 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

$$\overline{Q}(k)\underline{y}(1:(k+M)) = \begin{bmatrix} \beta\underline{u}(k) \\ \beta\underline{v}(k) \\ \underline{y}((k+1):(k+M)) \end{bmatrix}. \tag{44}$$

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

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

$$= \begin{bmatrix} \underline{u}(k+M) \\ \underline{v}(k+M) \end{bmatrix}, \tag{45}$$

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 RLS problem is

$$\underline{e}(1:(k+M)) = X(1:(k+M))\underline{w}(k+M) - \underline{y}(1:(k+M)), \tag{46}$$

and by the definition

$$\begin{aligned} E(1:(k+M)) &= ||X(1:(k+M))\underline{w}(k+M) - \underline{y}(1:k+M)||_2 \\ &= || \begin{bmatrix} R(k+M) \\ 0 \end{bmatrix} \underline{w}(k+M) - \begin{bmatrix} \underline{u}(k+M) \\ \underline{v}(k+M) \end{bmatrix} ||_2. \end{aligned} \tag{47}$$

To minimize the quantity $E(1:(k+M))$, the optimal weight vector is given by

$$R(k+M)\underline{w}_{opt}(k+M) = \underline{u}(k+M). \tag{48}$$

Then, substituting (48) into (46), the optimal residual vector becomes

$$\hat{e}(1:(k+M)) = \begin{bmatrix} 0 \\ -Q_2^H(k+M)v(k+M) \end{bmatrix}. \tag{49}$$

Therefore, the optimal residual vector for the time period from t_{N+1} to t_{k+M} is

$$\begin{aligned} \hat{e}((N+1):(k+M)) &= -Q_2^H(k+M)v(k+M) \\ &= -Q_2^H(k+M) \begin{bmatrix} \beta\underline{v}(k) \\ \alpha(k+M) \end{bmatrix}. \end{aligned} \tag{50}$$

The new CHT $Q(k+M)$ has the form

$$\begin{aligned} Q(k+M) &= \begin{bmatrix} Q_1(k+M) \\ Q_2(k+M) \end{bmatrix} \\ &= \begin{bmatrix} H_{11}(k+M)Q_1(k) & H_{12}(k+M) \\ Q_2(k) & 0 \\ H_{21}(k+M)Q_1(k) & H_{22}(k+M) \end{bmatrix}. \end{aligned} \tag{51}$$

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

$$Q_2(k+M) = \begin{bmatrix} Q_2(k) & 0 \\ H_{21}(k+M)Q_1(k) & H_{22}(k+M) \end{bmatrix}. \tag{52}$$

Substituting (52) into (50), we have

$$\begin{aligned} \hat{\underline{e}}((N+1):(k+M)) &= \begin{bmatrix} \hat{\underline{e}}((N+1):k) \\ \hat{\underline{e}}((k+1):(k+M)) \end{bmatrix} \\ &= -\begin{bmatrix} Q_2^H(k)\beta\underline{v}(k) + Q_1^H(k)H_{21}^H(k+M)\alpha(k+M) \\ H_{22}^H(k+M)\alpha(k+M) \end{bmatrix}, \end{aligned} \tag{53}$$

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

$$\hat{\underline{e}}((k+1):(k+M)) = -H_{22}^H(k+M)\alpha(k+M), \tag{54}$$

with

$$H_{22}^H = \prod_{i=1}^{N} H_{M\times M}^{(i)}, \tag{55}$$

and $H_{M\times M}^{(i)}$ is given in (15), i denotes the *ith* Householder transformation, and

$$\underline{e}((k+1):(k+M)) = \begin{bmatrix} e(t_{k+1}/(t_{k+1}:t_{k+M})) \\ e(t_{k+2}/(t_{k+1}:t_{k+M})) \\ \vdots \\ e(t_{k+M}/(t_{k+1}:t_{k+M})) \end{bmatrix}. \tag{56}$$

Compared to the SBHT-RLS algorithm in [22], 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 the entire data block to compute the residuals [22]. Thus, more data information is used for CHT-RLS than GR-RLS to compute the residuals. Specifically, from (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 time instant.

3.2 Systolic Array Implementation

The parallel CHT-RLS algorithm with fast initialization has been presented so far. We now consider the issue of VLSI systolic implementation. The systolic architecture for the parallel CHT-RLS algorithm is shown in Figure 5 which is able to obtain the residual immediately. However, it needs matrix-matrix multiplications in the boundary cells for which a large transmission bandwidth is required. Therefore, in Figure 7 a backward propagation array is added into the Figure 5 to avoid the matrix-matrix multiplication and be replaced by vector-vector multiplication as pointed out in [22]. In Figure 7, the delayed buffers are needed for each row to temporarily store parameters for vector multiplication. The boundary cell, internal cell, and the final cell are illustrated in Figure 6. The systolic algorithm for the boundary cell of the CHT-RLS system is shown in Table 3, the algorithm for the internal cell is shown in Table 4, and the algorithm for the final cell is described in Table 5. For the boundary cell described in Figure 5, it requires $2M^2+M$ multiplications, M^2+M addition, and one square root, while for the boundary cell described in Figure 7, it only requires $2M$ multiplications, M additions, and one square root where M is the block size. The internal cell shown in both figures requires $2M+5$ multiplications and $2M+1$ additions. The total counts of the whole systolic array has no meaning for parallel processing, since all the processor elements are computing at same time. It is useful to compute the maximum number of operations needed for each snapshot. For each snapshot, including boundary and internal cells, it requires $2M+5$ multiplications and $2M+1$ additions.

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

Table 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 4: The Algorithm for the Internal Cell of Systolic CHT-RLS

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

Table 5: The Algorithm for the Final Cell of Systolic CHT-RLS

4 Application to Array Processing

Adaptive arrays are currently the subject of extensive investigations for suppressions of the sidelobe interferences or jamming signals in many military radar, communications, and navigation systems [5, 6, 15, 25, 26, 27]. A modern adaptive array system is required to have rapid convergence, high cancellation performance, operational flexibility, and also be capable of achieving high throughput rate for real-time signal processing. For those applications it is suitable to build an open-loop recursive QRD-based systolic array system. Compared to the conventional adaptive array system, the QRD-based systolic array system not only improves the numerical accuracy but also increases the computational speed for updating the input signals. There has been much recent research work in designing QRD-based systolic arrays for sidelobe cancellation and adaptive antennas. However, most of the work have been based on Givens and modified Gram-Schmidt methods. Only until recently, systolic Householder-based RLS algorithm has been considered [21, 22]. In Figure 8, we propose a block diagram for sidelobe canceller. 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 8 that the output at the *ith* snapshot can be expressed as

$$e(i) = \sum_{l=1}^{N} x_l(i) w_l - y(i), \tag{57}$$

and the matrix form expression which is the same as (21) is given by

$$\underline{e}(n) = X(n)\underline{w}(n) - \underline{y}(n). \tag{58}$$

Therefore, the CHT-RLS systolic algorithm and architecture described in Section 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 Conclusions

In this paper, a recursive complex Householder algorithm with a fast initialization which can be programmed for both initialization and recursive computation is presented. Then a CHT-RLS algorithm with a systolic array processor is also proposed. Compared to the recursive block Householder algorithm described in [21, 22], the complex Householder algorithm saves $(M-1)N$ units of computational time for the initialization of the upper triangular matrix. In our proposed systolic architecture, the number of data snapshots needed for the initialization and the recursive updating is controlled by control codes that can be changed readily. Our CHT-RLS systolic architecture can be considered to be a generalization of McWhirter's QRD-RLS systolic array since it is well-known that Givens rotations is a special case of the Householder transformation [1]. The two-level pipelined implementation of the conventional HT has been developed. Similar to the two-level pipelined implementation based on the modified HT [22], the proposed scheme is also fully pipelinable. However, it does require one more multiply-and-add operation. The algorithm and architecture described in this paper appear promising for real-time array processing applications and VLSI hardware implementations.

References

[1] G. H. Golub and C. F. Van Loan, *Matrix Computation*, 2nd Ed. The Johns Hopkins University Press 1989.

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

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

[4] R. Schreiber, "Implementation of adaptive array algorithms," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-34, NO. 5, Oct. 1986, pp. 1038-1045.

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

[6] 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.

[7] 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.

[8] F. Ling, D. Manolakis, and J. G. Proakis, "A Recursive Modified Gram-Schmidt Algorithm for Least-Squares Estimation," *IEEE Trans. on ASSP*, vol. 34, 1986, pp. 829-836.

[9] S. Z. Kalson and K. Yao, "Systolic Array Processing for Order and Time Recursive Generalized Least-Squares Estimation," *Proc. SPIE*, Real-Time Signal Processing **VIII**, vol. 564, 1985, pp. 28-38.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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," *Proc. IEEE ICASSP 1991*, pp. 1033-1036.

[16] 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.

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

[18] C. M. Rader and A. O. Steinhardt, "Hyperbolic Householder Transformations." *IEEE Trans. on ASSP*, vol. ASSP-34, Dec. 1986, pp. 1589-1602.

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

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

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

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

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

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

[25] 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.

[26] S. M. Yuen, "Algorithmic, Architectural, and Beam Pattern Issues of Sidelobe Cancellation," *IEEE Trans. on AES*, vol. 25, no. 4, July 1989, pp. 459-472.

[27] 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, pp 862-867.

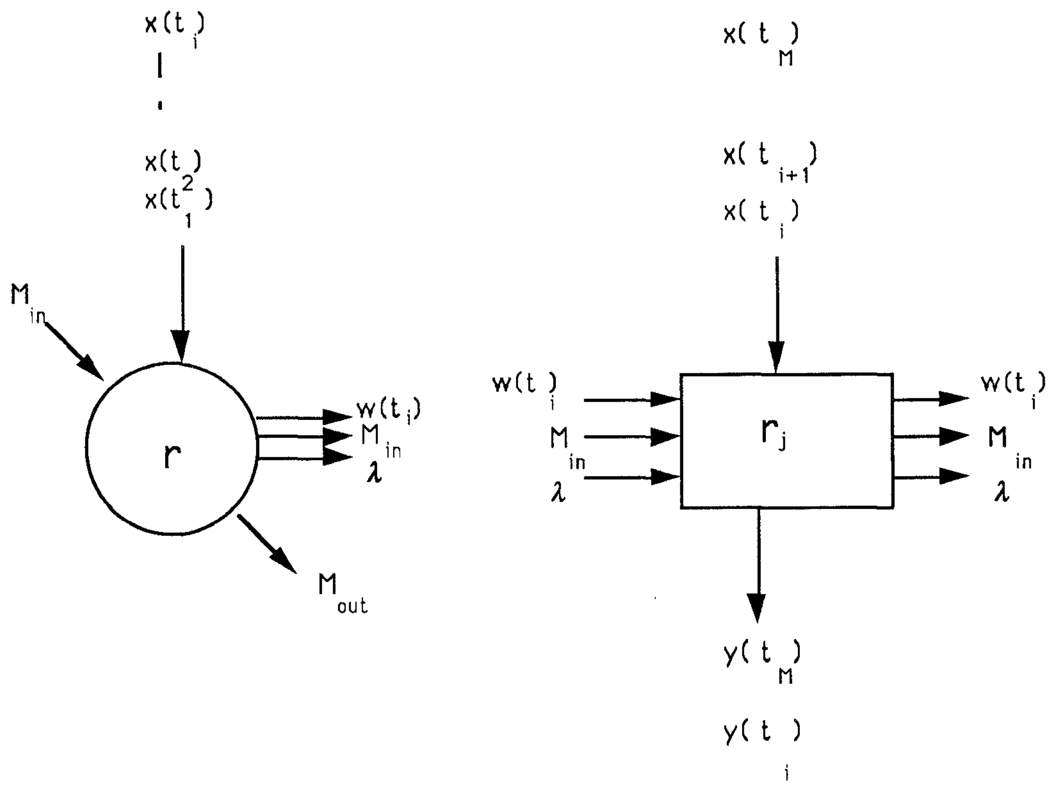

Figure 1: The Boundary and Internal Cells of Systolic Householder Transformation

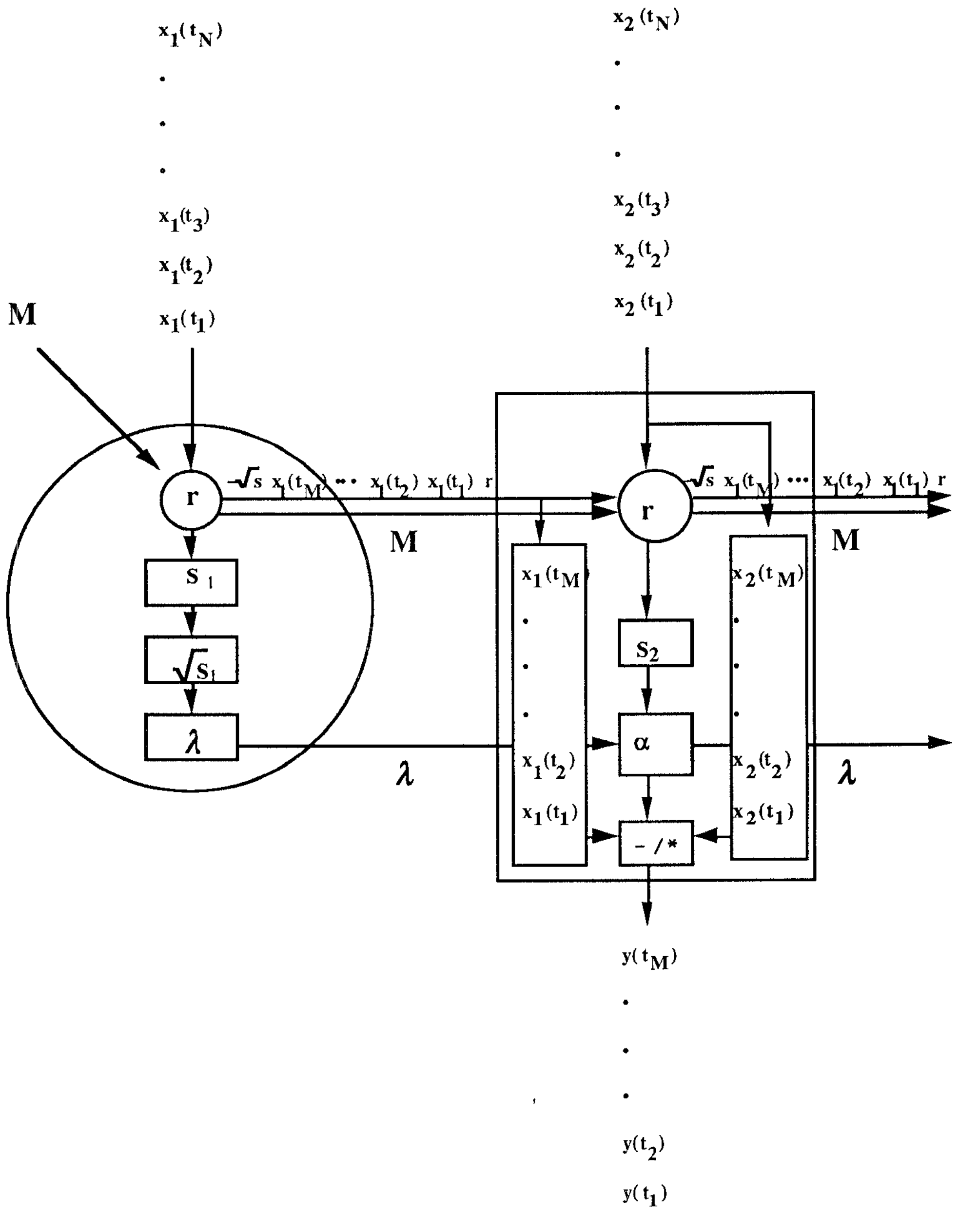

Figure 2: Processor Pair of Boundary and Internal Cells

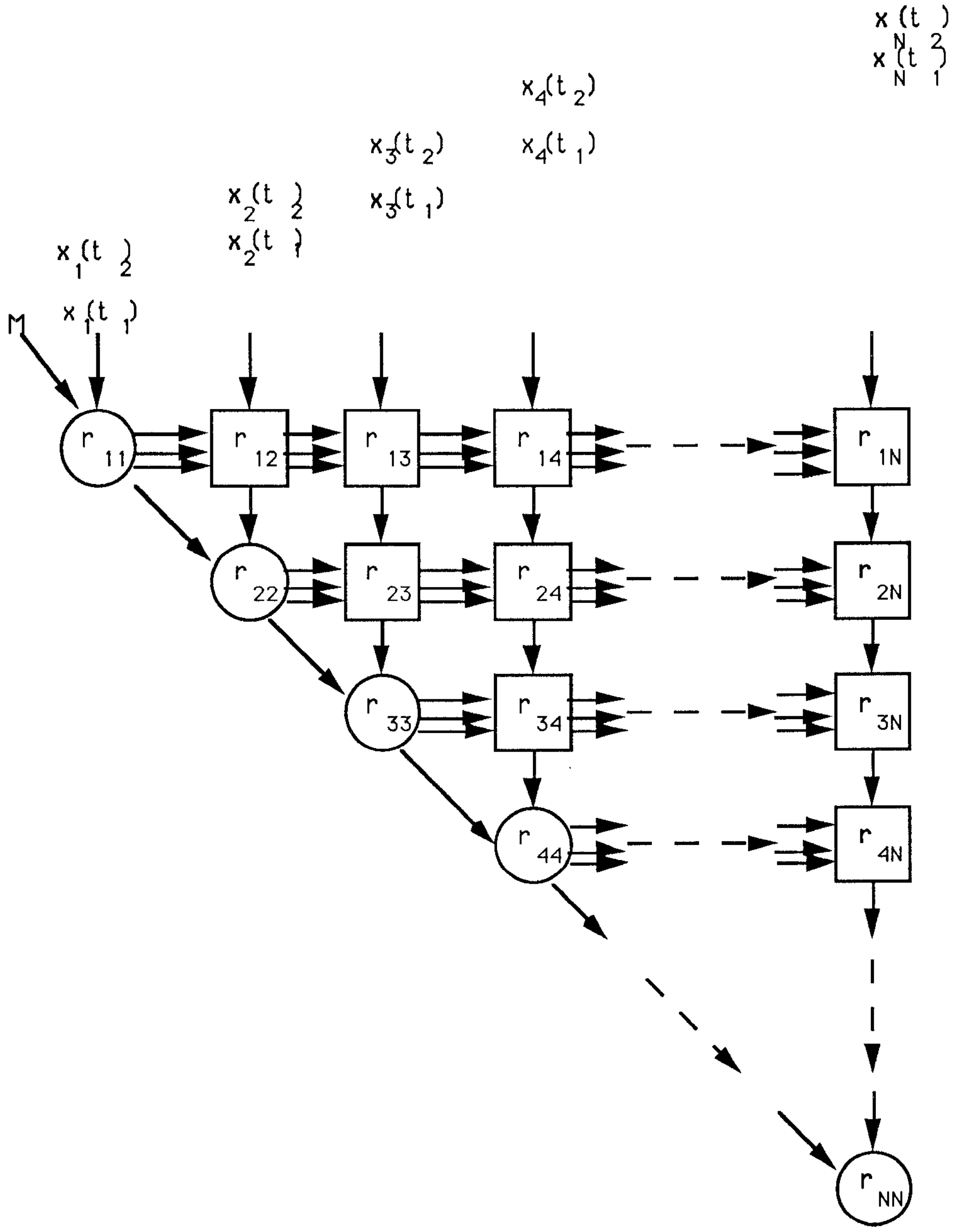

Figure 3: Triangular Systolic Architecture for Householder Transformation

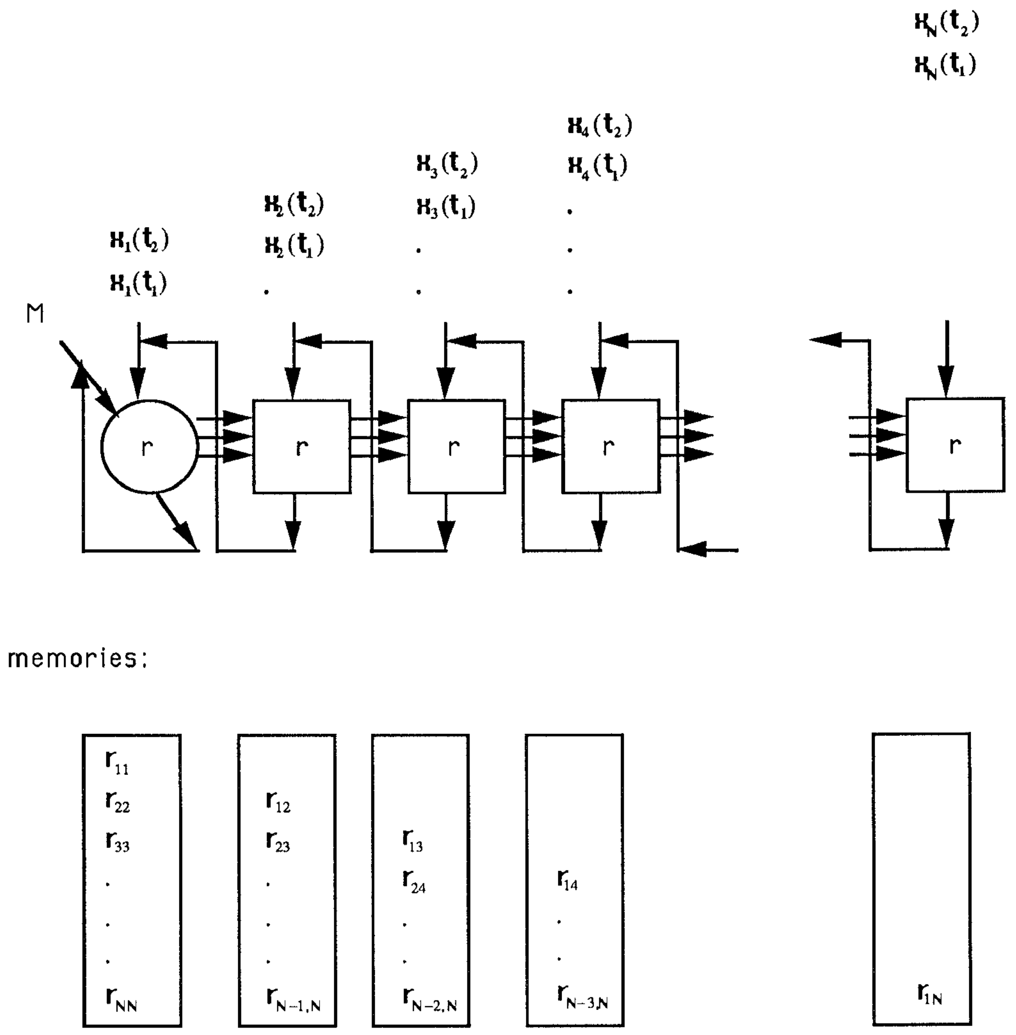

Figure 4: Linear Triangular Systolic Architecture for Householder Transformations with Feedback Configuration

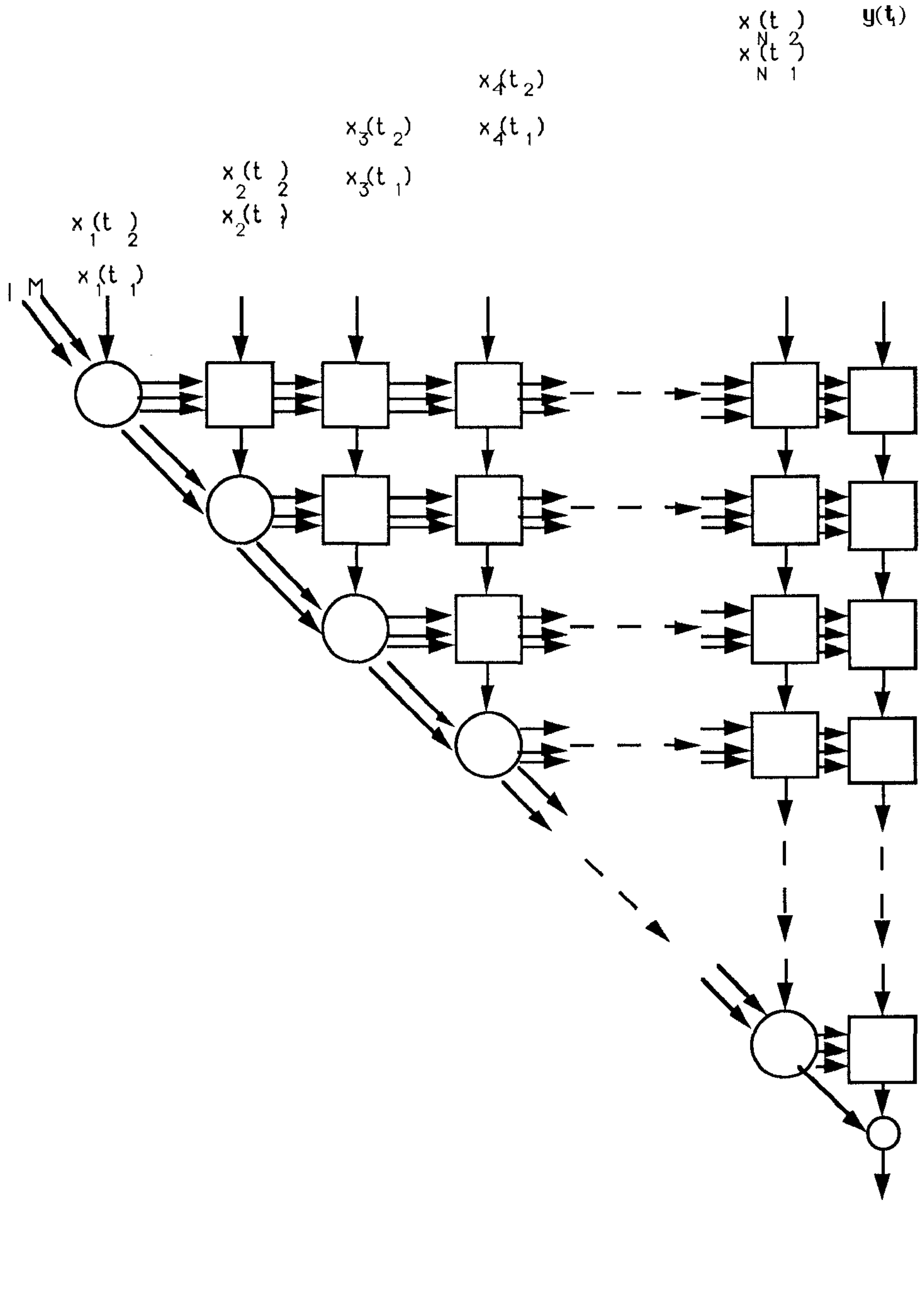

Figure 5: Systolic Architecture for CHT-RLS

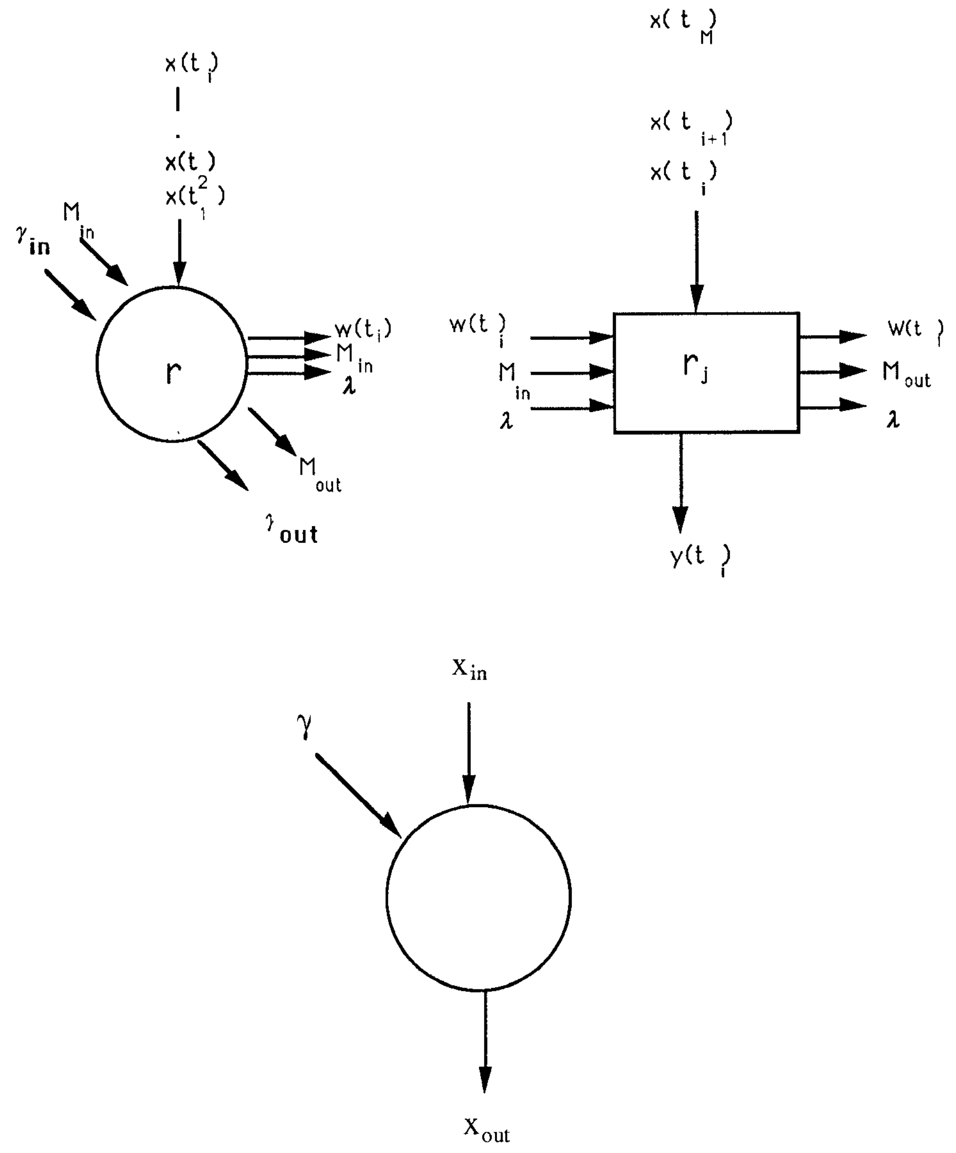

Figure 6: The Boundary, Internal, and Final Cells of Systolic CHT-RLS

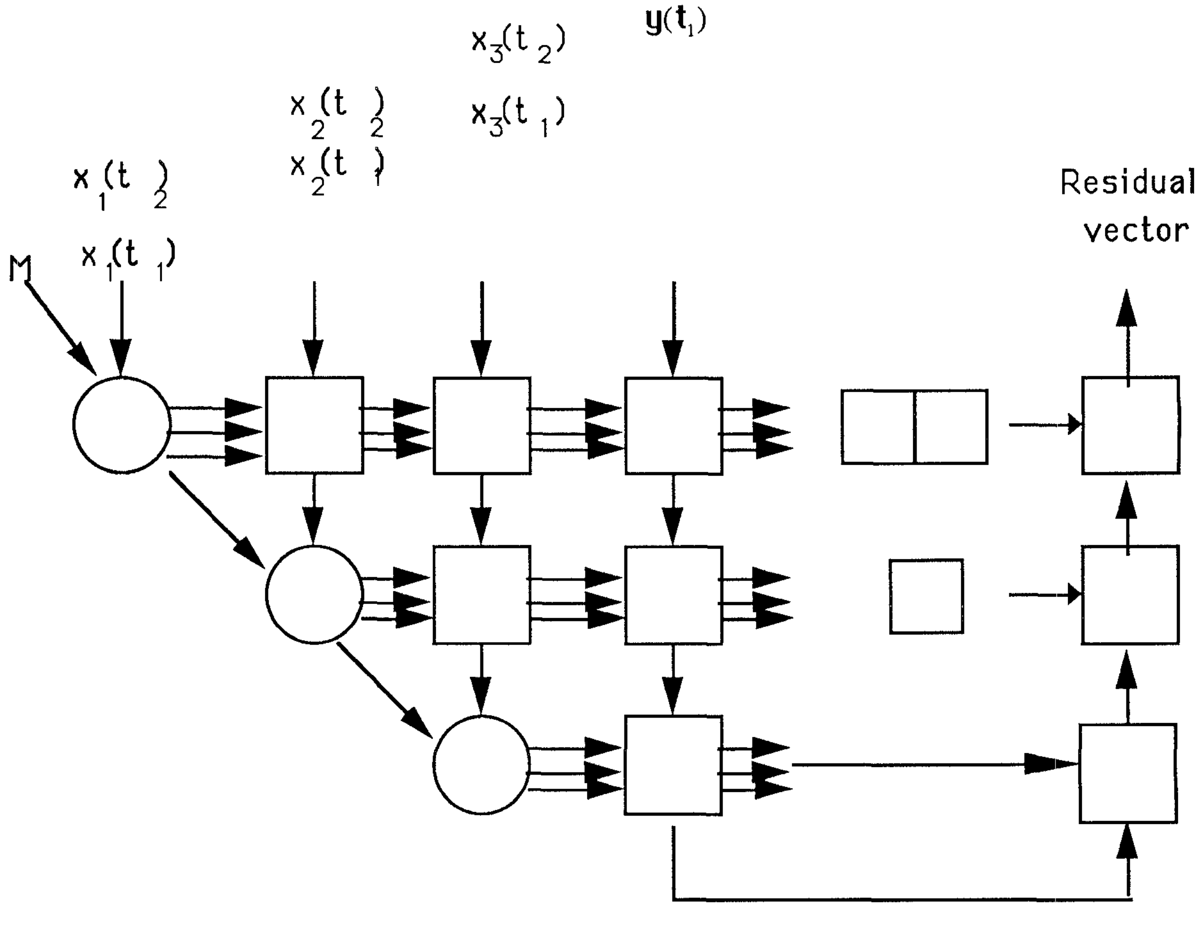

Figure 7: Systolic Architecture for CHT-RLS with Backward Propagation Array

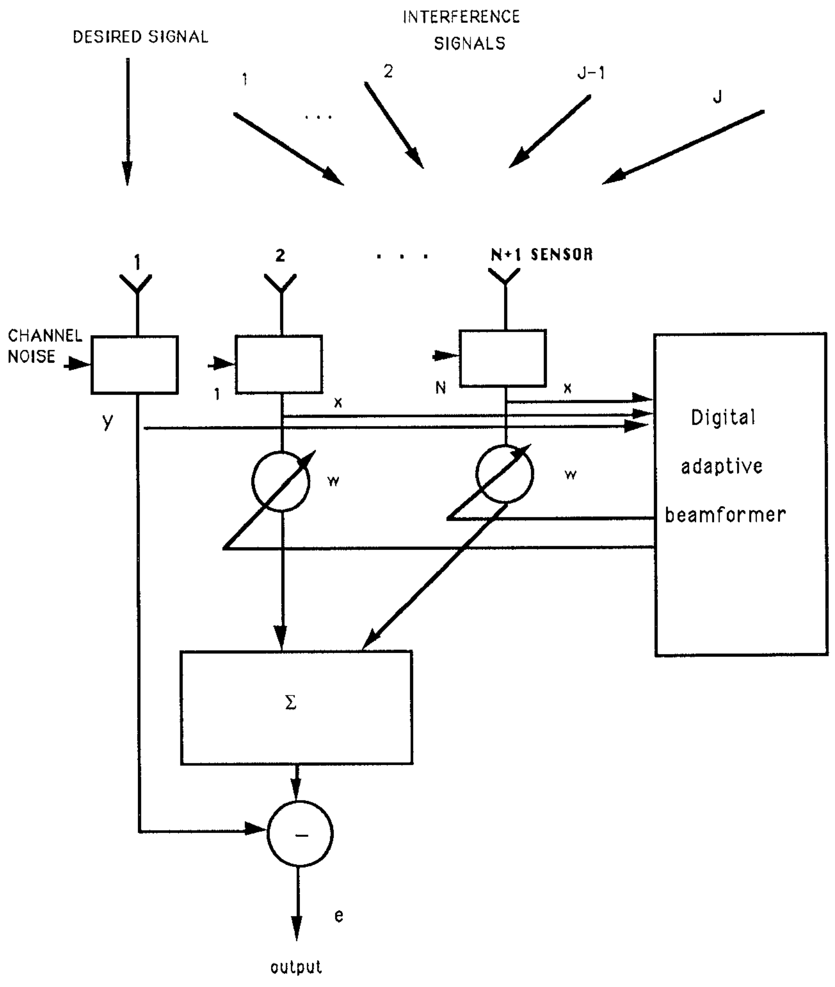

Figure 8: Sidelobe Cancellation Adaptive Beamforming System