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

# Systolic Block Householder Transformation for RLS Algorithm with Two-level Pipelined Implementation

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

TR 90-76

# Systolic Block Householder Transformation for RLS Algorithm with Two-level Pipelined Implementation

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

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

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

## ABSTRACT

The QRD RLS algorithm is one of the most promising RLS algorithms, due to its robust numerical stability and suitability for VLSI implementation based on a systolic array architecture. Up to now, among many techniques to implement the QR decomposition, only the Givens rotation and modified Gram-Schmidt methods have been successfully applied to the development of the QRD RLS systolic array. It is well-known that Householder transformation (HT) outperforms the Givens rotation method under finite precision computations. Presently, there is no known technique to implement the HT on a systolic array architecture. In this paper, we propose a *Systolic Block Householder Transformation* (SBHT) approach, to implement the HT on a systolic array as well as its application to the RLS algorithm. Since the data is fetched in a block manner, vector operations are in general required for the vectorized array. However, by using a modified HT algorithm, a two-level pipelined implementation can be used to pipeline the SBHT systolic array both at the vector and word levels. The throughput rate can be as fast as that of the Givens rotation method. Our approach makes the HT amenable for VLSI implementation as well as applicable to real-time high throughput applications of modern signal processing. The constrained RLS problem using the SBHT RLS systolic array is also considered in this paper.

**Submitted to IEEE Transaction on ASSP:** EDICS - 7.1.4 - Systolic and wavefront architectures.

This work is partially supported by UC MICRO grant and the NSF grant NCR-8814407.

## 1 Introduction

Least-squares (LS) technique constitutes an integral part of modern signal processing and communications methodology as used in adaptive filtering, beamforming, array signal processing, channel equalization, etc. [6]. Efficient implementation of the LS algorithm, particularly the recursive LS algorithm (RLS), is needed to meet the high throughput and speed requirements of modern signal processing. There are many possible approaches such as fast transversal method and lattice method which can perform RLS algorithm efficiently [1,6]. Unfortunately, these methods can encounter numerical difficulties due to the accumulation of round-off errors under a finite-precision implementation as summarized in [2]. This may lead to a divergence of the computations of the RLS algorithm [2]. A new type of systolic algorithm based on the QR decomposition (QRD) known as the QRD RLS was first proposed by McWhirter in [18]. This algorithm is one of the most promising algorithms in that it is numerical stable [1,12] as well as suitable for parallel processing implementation on a systolic array [6,18].

Up to now, most of the QRD RLS implementations were based on the Givens rotation method and modified Gram-Schmidt method which both are rank-1 update approaches [2,4, 7,13,16,18,9]. It is well-known that the Householder transformation (HT), which is a rank-$k$ update approach, is one of the most computationally efficient methods to compute QRD. The error analysis carried out by Wilkinson [25,8] showed that the HT outperforms the Givens method under finite precision computations. Presently, there is no known technique to implement the HT on a systolic array parallel processing architecture, since there is a belief that non-local connections in the implementation are necessary due to the vector processing nature of the Householder transformation. One of the purposes of this paper is to show that we can implement the HT on a systolic array with only local connections. Thus, it is amenable to VLSI implementation and is applicable to real-time high throughput applications of modern signal processing.

In this paper, we first propose a systolic Householder algorithm called a systolic block Householder transformation (SBHT) to compute the QRD with an implementation on a vectorized systolic array. Then a RLS algorithm based on the SBHT called SBHT RLS algorithm is proposed to perform RLS operations on the array. We shall show that the SBHT array and the SBHT RLS array are generalizations of Gentleman-Kung's QRD array [4] and McWhirter's QRD RLS systolic array [18] (see Fig.1) respectively. The difficulty in the applications of the above arrays is mainly due to the the vectorized operations of the processing cells. This results in a high cell complexity as well as a high I/O bandwidth. By using a modified HT algorithm proposed by Tsao [24], a two-level pipelined implementation of the SBHT RLS algorithm can be achieved. That is, the algorithm is pipelined at the vector level as well as at the word level. The complexity of the processing cell as well as the I/O bandwidth are thus reduced. In general, the cell complexity of the SBHT array is higher and the system latency is longer than that of the conventional Givens rotation implementations. With the two-level pipelined implementation, the throughput of the SBHT RLS systolic array is as fast as that of McWhirter's Givens rotation array, and it offers better numerical property than the Givens method. In addition, an extension of the SBHT RLS array to MVDR beamformation, which is a constrainted RLS problem, is also considered.

In section 2, a brief review of the QRD RLS algorithm is given. In section 3, the SBHT is presented while the SBHT RLS algorithm is considered in section 4. The two-level pipelined implementation of the SBHT RLS systolic array is discussed in section 5. Finally, in section 6, the constrained RLS problem, as applied to MVDR beamformation, using an extension of the SBHT RLS array is presented.

## 2 QRD RLS Algorithm

A full rank $m \times p$, $m > p$, rectangular matrix $\mathbf{X}$ can be uniquely factorized into two matrices $\mathbf{Q}$ and $\mathbf{R}$ such that $\mathbf{X} = \mathbf{QR}$, where $\mathbf{Q}$ is an $m \times p$ matrix with orthonormal columns and $\mathbf{R}$ is an $p \times p$ upper triangular matrix. Several different approaches of the QRD systolic arrays have been proposed by Gentleman and Kung [4], Heller and Ipsen [7], Luk [16], Ling, etc. [13], and Kalson and Yao [9]. The first three approaches are based on the Givens rotations methods, while the last two are based on the modified Gram-Schmidt orthogonalization. Given an $m \times 1$ vector $\mathbf{y}$, the LS problem is to minimize the norm of the residual vector $\underline{\epsilon}$

$$||\underline{\epsilon}(m)|| = ||\mathbf{X}(m)\mathbf{w}(m) - \mathbf{y}(m)||$$

by choosing an optimal weight vector $\mathbf{w}$. If the matrix $\mathbf{X}$ and vector $\mathbf{y}$ grow in time, then the problem of minimizing the norm of the residual vector recursively becomes the RLS problem. Until recently, it appears that only Givens and modified Gram-Schmidt methods have been considered for RLS computations. Some recent RLS problems based upon the use of Householder transformation have appeared [3,15]. In [18], McWhirter showed that a QRD RLS systolic array, which was based on the Gentleman-Kung's array, can be designed without first computing the weight vector of the RLS problem. This approach is useful for high throughput applications in various modern signal processing problems such as adaptive filtering and beamforming since optimal residuals are of direct interest while the weight vector needs not be computed. The basic idea of the QRD RLS systolic array in [18] is to update the $p \times p$ matrix $\mathbf{R}$ using a sequence of Givens rotation matrices when a new row of data arrives. Suppose we have the QRD of the data matrix $\mathbf{X}$ at time $m$ and expressed as $\mathbf{X}(m) = \mathbf{Q}(m)\mathbf{R}(m)$. Define

$$\bar{\mathbf{Q}}^T(m) = \left[\begin{array}{ccc} \mathbf{Q}^T(m) & \vdots & \mathbf{0} \\ ---- & & ---- \\ \mathbf{0^T} & \vdots & 1 \end{array}\right].$$

When a new row of data arrives, we then have

$$\bar{\mathbf{Q}}^T(m)\mathbf{A}(m+1) = \left[\begin{array}{c} \mathbf{R}(m) \\ ---- \\ \mathbf{0} \\ ---- \\ x_1, x_2, \cdots, x_p \end{array}\right].$$

This new row of data can be zeroed out by applying a sequence of Givens rotations

$$\mathbf{G} = \mathbf{G}_p \cdots \mathbf{G}_2 \mathbf{G}_1$$

where the $(m+1) \times (m+1)$ transformation matrix $\mathbf{G}_i$ is defined by

$$\mathbf{G}_i = \begin{bmatrix} \mathbf{I}_{i-1} & \vdots & 0 & \vdots & \mathbf{0} & \vdots & 0 \\ --- & & & & --- & & \\ 0 & & c_i & & 0 & & s_i \\ --- & & & & --- & & \\ \mathbf{0} & \vdots & 0 & \vdots & \mathbf{I}_{m-i} & \vdots & 0 \\ --- & & & & --- & & \\ 0 & \vdots & -s_i & \vdots & 0 & \vdots & c_i \end{bmatrix},$$

with

$$c_i = \frac{a}{\sqrt{a^2+b^2}}, \quad s_i = \frac{b}{\sqrt{a^2+b^2}},$$

where $a$ and $b$ are elements of vectors in the $i^{th}$ and $(m+1)^{th}$ rows under rotation.

Fig.1a shows the systolic array proposed by McWhirter in [18]. It consists of a QRD triarray and a linear response array (RA). The rotation parameters are propagated from the boundary cells to the right for internal cells to update their contents, and the cosine parameters are also cumulated and propagated down diagonal boundary cells. Each skewed input row of data is zeroed out by the QRD triarray. The optimal residual is then obtained by the multiplication of the cumulated cosines and the rotated output of the desired response at the response array (see Fig.1a) [18].

## 3 Systolic Block Householder Transformation

The Givens rotation method discussed above is a rank-1 update approach since each input consists of one row of data. For the systolic block Householder transformation (SBHT), we need a block data formulation. Denote the *data matrix* as

$$\mathbf{X}(n) = \begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \vdots \\ \mathbf{X}_n^T \end{bmatrix} = \begin{bmatrix} \mathbf{X}(n-1) \\ --- \\ \mathbf{X}_n^T \end{bmatrix} \in \Re^{nk \times p} \tag{1}$$

and the *desired response vector* as

$$\mathbf{y}(n) = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} \mathbf{y}(n-1) \\ --- \\ \mathbf{y}_n \end{bmatrix} \in \Re^{nk}, \tag{2}$$

where $\mathbf{X}_i^T$ is the $k \times p$ $i^{th}$ data block matrix,

$$\mathbf{X}_i^T = \begin{bmatrix} \mathbf{x}_{(i-1)k+1}^T \\ \mathbf{x}_{(i-1)k+2}^T \\ \vdots \\ \mathbf{x}_{ik}^T \end{bmatrix} = [\mathbf{x}_{i,1}\ \mathbf{x}_{i,2} \cdots \mathbf{x}_{i,p}] \tag{3}$$

$$= \begin{bmatrix} x_{(i-1)k+1,1} & x_{(i-1)k+1,2} & \cdots & x_{(i-1)k+1,p} \\ x_{(i-1)k+2,1} & x_{(i-1)k+2,2} & \cdots & x_{(i-1)k+2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{ik,1} & x_{ik,2} & \cdots & x_{ik,p} \end{bmatrix} \in \Re^{k\times p} \tag{4}$$

and $\mathbf{y}_i$ is the $k \times 1$ $i^{th}$ desired response block vector.

$$\mathbf{y}_i = \begin{bmatrix} y_{(i-1)k+1} \\ y_{(i-1)k+2} \\ \vdots \\ y_{ik} \end{bmatrix} \in \Re^{k}. \tag{5}$$

$k$ is the *block size* and $p$ is the *order*(i.e., number of columns) of the system.

For a rank-$k$ update QR decomposition, suppose we have

$$\mathbf{Q}(n-1)\mathbf{X}(n-1) = \begin{bmatrix} \mathbf{R}(n-1) \\ --- \\ 0 \end{bmatrix}. \tag{6}$$

Denote

$$\bar{\mathbf{Q}}_k(n-1) = \begin{bmatrix} \mathbf{Q}(n-1) & \vdots & \mathbf{0} \\ --- & & --- \\ \mathbf{0}^T & \vdots & \mathbf{I}_k \end{bmatrix}, \tag{7}$$

then we have

$$\bar{\mathbf{Q}}_k(n-1)\mathbf{X}(n) = \begin{bmatrix} \mathbf{R}(n-1) \\ --- \\ \mathbf{0} \\ --- \\ \mathbf{X}_n^T \end{bmatrix}. \tag{8}$$

If we can find a matrix $\mathbf{H}(n)$ such that

$$\mathbf{H}(n)\bar{\mathbf{Q}}_k(n-1)\mathbf{X}(n) = \begin{bmatrix} \mathbf{R}(n) \\ --- \\ 0 \end{bmatrix}, \tag{9}$$

then the new $\mathbf{Q}(n)$ is

$$\mathbf{Q}(n) = \mathbf{H}(n)\bar{\mathbf{Q}}_k(n-1). \tag{10}$$

An $n \times n$ Householder transformation matrix $\mathbf{T}$ is of the form

$$\mathbf{T} = \mathbf{I} - \frac{2\mathbf{z}\mathbf{z}^{\mathbf{T}}}{||\mathbf{z}||^2}, \tag{11}$$

where $\mathbf{z} \in \Re^n$ [5]. When a vector $\mathbf{x}$ is multiplied by $\mathbf{T}$, it is reflected in the hyperplane defined by $span\{\mathbf{z}\}^\perp$. Choosing $\mathbf{z} = \mathbf{x} \pm ||\mathbf{x}||_2\mathbf{e}_1$, where $\mathbf{e}_1 = [1, 0, 0, \cdots, 0] \in \Re^n$, then $\mathbf{x}$ is reflected onto $\mathbf{e}_1$ by $\mathbf{T}$ as

$$\mathbf{T}\mathbf{x} = \pm||\mathbf{x}||_2\mathbf{e}_1. \tag{12}$$

That is, all of the energy of $\mathbf{x}$ is reflected onto unit vector $\mathbf{e}_1$ after the transformation. We can zero out $\mathbf{X}_n^T$ by applying successive Householder transformations as follows,

$$\mathbf{H}^{(i)}(n)\begin{bmatrix} \mathbf{R}^{(i-1)}(n-1) \\ -------- \\ \mathbf{0} \\ -------- \\ 0,\cdots,0,\mathbf{x}_{n,i}^{(i-1)},\cdots,\mathbf{x}_{n,p}^{(i-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{(i)}(n-1) \\ -------- \\ \mathbf{0} \\ -------- \\ 0,\cdots,0,0,\mathbf{x}_{n,i+1}^{(i)},\cdots,\mathbf{x}_{n,p}^{(i)} \end{bmatrix},$$

for $i = 1,\cdots,p$, where $\mathbf{x}_{n,i}^{(0)} = \mathbf{x}_{n,i}$, $\mathbf{R}^{(0)}(n-1) = \mathbf{R}(n-1)$, and the resultant matrix $\mathbf{H}(n)$ is

$$\mathbf{H}(n) = \mathbf{H}^{(p)}(n)\mathbf{H}^{(p-1)}(n)\cdots\mathbf{H}^{(1)}(n), \tag{13}$$

where each $\mathbf{H}^{(i)}(n)$ represents a Householder transformation which zeros out the $i^{th}$ column of the updated $\mathbf{X}_n^T$, *i.e.*, $\mathbf{x}_{n,i}^{(i-1)}$.

To obtain $\mathbf{H}^{(1)}(n)$, denote

$$\mathbf{z}_1 = \begin{bmatrix} r_{11}-\sigma_1 & \vdots & \mathbf{0}_{(n-1)k-1}^T & \vdots & \mathbf{x}_{n,1}^T \end{bmatrix}^T,$$

where $r_{11}$ is the $(1,1)$ element of $\mathbf{R}(n-1)$, $\sigma_1^2 = r_{11}^2 + ||\mathbf{x}_{n,1}||^2$. Then from (11)

$$\mathbf{H}^{(1)}(n) = \begin{bmatrix} h_{11}^{(1)}(n) & \vdots & \mathbf{0}^T & \vdots & \mathbf{h}_{12}^{(1)T}(n) \\ -- & & ---- & & --- \\ \mathbf{0} & \vdots & I_{(n-1)k-1} & \vdots & \mathbf{0} \\ -- & & ---- & & --- \\ \mathbf{h}_{21}^{(1)}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{22}^{(1)}(n) \end{bmatrix}, \tag{14}$$

where $h_{11}^{(1)}(n)$ is a scalar, $\mathbf{h}_{12}^{(1)}(n)$ is a $k\times 1$ vector, $\mathbf{h}_{21}^{(1)}(n) = \mathbf{h}_{12}^{(1)}(n)$, and $\mathbf{H}_{22}^{(1)}(n)$ is a $k\times k$ matrix given by

$$\mathbf{H}_{22}^{(1)}(n) = \mathbf{I}_k - \frac{2\mathbf{x}_{n,1}\mathbf{x}_{n,1}^T}{\sigma_{\mathbf{z}_1}^2}, \tag{15}$$

with $\sigma_{\mathbf{z}_1}^2 = ||\mathbf{z}_1||_2^2 = 2(\sigma_1^2 - \sigma_1 r_{11})$. Define $\psi_1 = \sigma_1^2 - \sigma_1 r_{11}$, (15) can be rewritten in a form without multiplication of the number 2 as

$$\mathbf{H}_{22}^{(1)}(n) = \mathbf{I}_k - \frac{\mathbf{x}_{n,1}\mathbf{x}_{n,1}^T}{\psi_1}.$$

In general,

$$\mathbf{H}^{(m)}(n) = \begin{bmatrix} \mathbf{H}_{11}^{(m)}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{12}^{(m)}(n) \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{0} & \vdots & \mathbf{I}_{(n-1)k-p} & \vdots & \mathbf{0} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{H}_{21}^{(m)}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{22}^{(m)}(n) \end{bmatrix}, \tag{16}$$

where $\mathbf{H}_{11}^{(m)}(n) \in \Re^{p\times p}$ is an identity matrix except for the $m^{th}$ diagonal entry; $\mathbf{H}_{12}^{(m)}(n) \in \Re^{p\times k}$ is a zero matrix except for the $m^{th}$ row; $\mathbf{H}_{21}^{(m)}(n) = \mathbf{H}_{12}^{(m)T}(n)$; and

$$\mathbf{H}_{22}^{(m)}(n) = \mathbf{I}_k - \frac{\mathbf{x}_{n,m}^{(m-1)}\mathbf{x}_{n,m}^{(m-1)T}}{\psi_m} \in \Re^{k\times k} \tag{17}$$

is symmetric with $\psi_m = \sigma_m^2 - \sigma_m r_{mm}$, where $\sigma_m^2 = r_{mm}^2 + \|\mathbf{x}_{n,m}^{(m-1)}\|^2$. It can be easily seen that $\mathbf{H}_{12}^{(i)}(n)\mathbf{H}_{21}^{(j)}(n) = \mathbf{0}, \mathbf{H}_{21}^{(i)}(n)\mathbf{H}_{12}^{(j)}(n) = \mathbf{0}$, for $\forall i \neq j$. Thus we have the following lemma,

**Lemma 1:** The Householder transformation matrix, $\mathbf{H}(n) \in \Re^{nk\times nk}$, is orthogonal and is of the form

$$\mathbf{H}(n) = \begin{bmatrix} \mathbf{H}_{11}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{12}(n) \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{0} & \vdots & \mathbf{I}_{(n-1)k-p} & \vdots & \mathbf{0} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{H}_{21}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{22}(n) \end{bmatrix} = \mathbf{H}^{(p)}(n)\mathbf{H}^{(p-1)}(n)\cdots\mathbf{H}^{(1)}(n), \tag{18}$$

with

$$\begin{aligned} \mathbf{H}_{11}(n) &= \mathbf{H}_{11}^{(p)}(n)\cdots\mathbf{H}_{11}^{(2)}(n)\mathbf{H}_{11}^{(1)}(n) \\ \mathbf{H}_{22}(n) &= \mathbf{H}_{22}^{(p)}(n)\cdots\mathbf{H}_{22}^{(2)}(n)\mathbf{H}_{22}^{(1)}(n).\square \end{aligned} \tag{19}$$

For the block size of $k = 1$, then the Givens rotation method reduces to the special case of the rank-1 update Householder transformation [5], and the $\mathbf{H}$ matrix in *Lemma 1* becomes a Givens rotation matrix $\mathbf{G}$ of the form [18]

$$\mathbf{G}(n) = \begin{bmatrix} \mathbf{K}(n) & | & \mathbf{0} & | & \mathbf{h}(n) \\ ---- & & ---- & & ---- \\ \mathbf{0} & | & \mathbf{I}_{n-p-1} & | & \mathbf{0} \\ ---- & & ---- & & ---- \\ \mathbf{h}^H(n) & | & \mathbf{0} & | & \gamma(n) \end{bmatrix}$$

where $\mathbf{K}(n)$ is a $p \times p$ matrix, $\mathbf{h}(n)$ is a $p \times 1$ vector, and $\gamma(n)$ is a scalar given by $\gamma(n) = \prod_{i=1}^{p} c_i(n),\ \ n \geq p$ where $c_i(n)$ is the cosine parameter associated with the $i^{th}$ Given rotation.

## 3.1 Vectorized SBHT QRD Systolic Array

Now we propose a vectorized systolic array to implement the QRD based on the SBHT. Similar to the QR triarray of Gentleman-Kung [4], this array has both boundary and internal cells. The boundary cell takes an input of block size $k$ from the above internal processor or directly from the input port, updates its content and generates the reflection vector, and sends it to the right for the internal cell processing (see Fig.2a). Define

$$\bar{\mathbf{x}}_{n,i}^{(i-1)^T} = \begin{bmatrix} \mathbf{0}_{i-1} & \vdots & r_{ii} & \vdots & \mathbf{0}_{(n-1)k-i} & \vdots & \mathbf{x}_{n,i}^{(i-1)^T} \end{bmatrix},\quad i = 1, \cdots, p,$$

and $\mathbf{z}_i = \bar{\mathbf{x}}_{n,i}^{(i-1)} - \sigma_i \mathbf{e}_i$, where $\mathbf{e}_i$ is a zero vector except for a unity at the $i^{th}$ position. When an internal cell receives the reflection vector, instead of forming the matrix $\mathbf{z}_i\mathbf{z}_i^T$ and performing matrix arithmetics, it performs an inner product operation to update its content $r_{ii}$ by doing

$$\mathbf{H}^{(i)}(n)\bar{\mathbf{x}}_{n,j}^{(i-1)} = \bar{\mathbf{x}}_{n,j}^{(i-1)} - \frac{\mathbf{z}_i}{\psi_i}(\mathbf{z}_i^T \cdot \bar{\mathbf{x}}_{n,j}^{(i-1)}), \quad j = i+1, \ldots, p, \tag{20}$$

and sends the reflected data $\mathbf{x}_{n,j}$ downward for further processing. Fig.2 shows the SBHT QRD array architecture and the processing cells. When the block size is $k = 1$, this vectorized array degenerates to the Gentleman-Kung's Givens rotation triarray.

## 4 SBHT RLS Algorithm

The LS problem is to choose a *weight vector* $\mathbf{w}(n) \in \Re^p$, such that the *block-forgetting norm* of

$$\underline{\epsilon}(n) = \begin{bmatrix} \mathbf{e}_1(n) \\ \mathbf{e}_2(n) \\ \vdots \\ \mathbf{e}_n(n) \end{bmatrix} = \mathbf{X}(n)\mathbf{w}(n) - \mathbf{y}(n) \tag{21}$$

is minimized. The optimal weight vector $\hat{\mathbf{w}}(n)$ satisfies

$$\min_w ||\underline{\epsilon}(n)||_{\Lambda_k} = ||\mathbf{X}(n)\hat{\mathbf{w}}(n) - \mathbf{y}(n)||_{\Lambda_k}, \tag{22}$$

where

$$||\underline{\epsilon}(n)||_{\Lambda_k} = ||\mathbf{\Lambda}_k(n)\underline{\epsilon}(n)||_2 = \sqrt{\sum_{i=1}^{n} \lambda^{2(n-i)} \cdot ||\mathbf{e}_i(n)||_2^2}, \qquad 0 < \lambda \leq 1 \tag{23}$$

$\mathbf{\Lambda}_k(n)$ is a block-diagonal exponential weighting matrix of the form,

$$\mathbf{\Lambda}_k(n) = \begin{bmatrix} \lambda^{n-1}\mathbf{I}_k & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \lambda\mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I}_k \end{bmatrix} \in \Re^{nk \times nk} \tag{24}$$

and $|| \cdot ||_2$ is the Euclidean norm,

$$||\mathbf{e}_i(n)||_2^2 = \sum_{j=1}^{k} |e_{(i-1)k+j}(n)|^2. \tag{25}$$

The exponential forgetting weighting $\lambda$ is incorporated in the RLS filtering scheme to avoid overflow in the processors as well as to facilitate nonstationary data updating.

The QRD of the *weighted augmented data matrix* at time $n$ (in the block sense, it is equivalent to $nk$ snapshots), is given by

$$\mathbf{\Lambda}_k(n)\left[\mathbf{X}(n) \vdots \mathbf{y}(n)\right] = [\mathbf{Q}_1^T(n) \vdots \mathbf{Q}_2^T(n)] \begin{bmatrix} \mathbf{R}(n) & \vdots & \mathbf{u}(n) \\ \mathbf{0} & \vdots & \mathbf{v}(n) \end{bmatrix}, \tag{26}$$

where

$$\mathbf{Q}(n) = \begin{bmatrix} \mathbf{Q}_1(n) \\ --- \\ \mathbf{Q}_2(n) \end{bmatrix},$$

constitutes an orthogonal transformation matrix with $\mathbf{Q}_1(n) \in \Re^{p\times nk}$ spanning the column space of the weighted data matrix $\mathbf{\Lambda}_k(n)\mathbf{X}(n)$ and $\mathbf{Q}_2(n) \in \Re^{(nk-p)\times nk}$ spanning the null space, $\mathbf{R}(n) \in \Re^{p\times p}$ is an upper triangular matrix and

$$\mathbf{Q}(n)\mathbf{y}(n) = \begin{bmatrix} \mathbf{u}(n) \\ \mathbf{v}(n) \end{bmatrix}.$$

The optimal weight vector can be obtained by solving

$$\mathbf{R}(n)\hat{\mathbf{w}}(n) = \mathbf{u}(n). \tag{27}$$

Obviously, $\mathbf{\Lambda}_k(n)\mathbf{X}(n) = \mathbf{Q}_1^T(n)\mathbf{R}(n)$. As a result, the weighted optimal residual of (21) is,

$$\begin{aligned} \mathbf{\Lambda}_k(n)\hat{\underline{e}}(n) &= \mathbf{Q}_1^T(n)\mathbf{R}(n)\hat{\mathbf{w}}(n) - \mathbf{Q}_1^T(n)\mathbf{u}(n) - \mathbf{Q}_2^T(n)\mathbf{v}(n) \\ &= -\mathbf{Q}_2^T(n)\mathbf{v}(n), \end{aligned} \tag{28}$$

which lies in the null space of the weighted data matrix.

Now, suppose we have the data matrix up to time $n-1$ and the QRD of $\mathbf{\Lambda}_k(n-1)[\mathbf{X}(n-1) \vdots \mathbf{y}(n-1)]$, then the recursive LS problem is to efficiently compute the optimum residual at time $n$ from the results we have at time $n-1$. In particular, we are interested in the new $n^{th}$ block of the optimal residual,

$$\hat{\mathbf{e}}_n(n) = \mathbf{X}_n^T\hat{\mathbf{w}}(n) - \mathbf{y}_n. \tag{29}$$

From (8), (9) and (18), (26) can be expressed as

$$\begin{bmatrix} \mathbf{R}(n) & \vdots & \mathbf{u}(n) \\ \mathbf{0} & \vdots & \mathbf{v}(n) \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{11}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{12}(n) \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{0} & \vdots & \mathbf{I}_{(n-1)k-p} & \vdots & \mathbf{0} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{H}_{21}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{22}(n) \end{bmatrix} \cdot \begin{bmatrix} \lambda\mathbf{R}(n-1) & \vdots & \lambda\mathbf{u}(n-1) \\ ---- & & ---- \\ \mathbf{0} & \vdots & \lambda\mathbf{v}(n-1) \\ ---- & & ---- \\ \mathbf{X}_n^T & \vdots & \mathbf{y}_n \end{bmatrix}.$$

By recursion on $n$, we relate $\mathbf{Q}(n)$ and $\mathbf{Q}(n-1)$ using (10) and have

$$\mathbf{Q}(n) = \begin{bmatrix} \mathbf{H}_{11}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{12}(n) \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{0} & \vdots & \mathbf{I}_{(n-1)k-p} & \vdots & \mathbf{0} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{H}_{21}(n) & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{22}(n) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{Q}_1(n-1) & \vdots & \mathbf{0} \\ --- & & --- \\ \mathbf{Q}_2(n-1) & \vdots & \mathbf{0} \\ --- & & --- \\ \mathbf{0} & \vdots & \mathbf{I}_k \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{H}_{11}(n)\mathbf{Q}_1(n-1) & \vdots & \mathbf{H}_{12}(n) \\ -------- & & --- \\ \mathbf{Q}_2(n-1) & \vdots & \mathbf{0} \\ -------- & & --- \\ \mathbf{H}_{21}(n)\mathbf{Q}_1(n-1) & \vdots & \mathbf{H}_{22} \end{bmatrix}. \tag{30}$$

We can see that $\mathbf{Q}_2(n)$ is updated from $\mathbf{Q}_1(n-1)$ and $\mathbf{Q}_2(n-1)$ by

$$\mathbf{Q}_2(n) = \begin{bmatrix} \mathbf{Q}_2(n-1) & \vdots & \mathbf{0} \\ -------- & & --- \\ \mathbf{H}_{21}(n)\mathbf{Q}_1(n-1) & \vdots & \mathbf{H}_{22} \end{bmatrix}. \tag{31}$$

On the other hand, the updated $[\mathbf{u}^T(n), \mathbf{v}^T(n)]^T$ is

$$\begin{aligned} \begin{bmatrix} \mathbf{u}(n) \\ --- \\ \mathbf{v}(n) \end{bmatrix} &= \mathbf{Q}(n)\mathbf{\Lambda}_k(n) \begin{bmatrix} \mathbf{y}(n-1) \\ --- \\ \mathbf{y}_n \end{bmatrix} \\ &= \begin{bmatrix} \lambda\mathbf{H}_{11}(n)\mathbf{u}(n-1) + \mathbf{H}_{12}(n)\mathbf{y}_n \\ ------------- \\ \lambda\mathbf{v}(n-1) \\ ------------- \\ \mathbf{v}_n \end{bmatrix}, \end{aligned} \tag{32}$$

where

$$\mathbf{v}_n = \lambda\mathbf{H}_{21}(n)\mathbf{u}(n-1) + \mathbf{H}_{22}(n)\mathbf{y}_n. \tag{33}$$

Therefore, from (28), (31), and (32), the weighted optimal residual vector can be obtained from parameters at time $n-1$ by

$$\mathbf{\Lambda}_k(n)\underline{\hat{\epsilon}}(n) = \begin{bmatrix} \underline{\hat{\epsilon}}(n-1|n) \\ --- \\ \hat{\mathbf{e}}_n(n) \end{bmatrix} = \begin{bmatrix} -\lambda\mathbf{Q}_2^T(n-1)\mathbf{v}(n-1) - \mathbf{Q}_1^T(n-1)\mathbf{H}_{21}^T(n)\mathbf{v}_n \\ -------------------- \\ -\mathbf{H}_{22}^T(n)\mathbf{v}_n \end{bmatrix}, \tag{34}$$

where $\underline{\hat{\epsilon}}(m|n)$ denotes the estimate of $\underline{\hat{\epsilon}}$ at time $m$, $m \leq n$, given all of the data up to time $n$. The new $n^{th}$ block of the optimal residual is then obtained as

$$\hat{\mathbf{e}}_n(n) = -\mathbf{H}_{22}^T(n)\mathbf{v}_n = -\mathbf{H}_{22}^{(1)}(n)\mathbf{H}_{22}^{(2)}(n)\cdots\mathbf{H}_{22}^{(p)}(n)\mathbf{v}_n. \tag{35}$$

For the block size of $k=1$, all vector parameters in (35) become scalars and can be expressed as

$$e_n(n) = -\prod_{i=1}^{p} c_i v_n, \tag{36}$$

which was first shown by McWhirter in [18]. Note that there are some differences between the optimal residuals estimated by SBHT and Givens rotation methods. To be specific, the optimal residual vector in (35) is given by

$$\hat{\mathbf{e}}_n(n) = \begin{bmatrix} e_{(n-1)k+1}((n-1)k+1|nk) \\ \vdots \\ e_{nk-1}(nk-1|nk) \\ e_{nk}(nk|nk) \end{bmatrix}, \tag{37}$$

while, the optimal residual estimated by the Givens rotation method in (36) is

$$\hat{e}_n(n) = e_n(n|n). \tag{38}$$

In this sense, the SBHT RLS gives a better estimate of the residual since it uses more data samples to estimate the optimal residual. As an example, consider $k = 2$. Then the optimal residuals obtained from the SBHT RLS and Givens methods are $[e_{2n-1}((2n-1)|2n), e_{2n}(2n|2n)]$ and $[e_{2n-1}((2n-1)|(2n-1), e_{2n}(2n|2n)]$ respectively. It is clear now that the SBHT RLS method gives a better estimate for the previous residual than the Givens rotation method because the former makes use of the future data sample at time $2n$ to estimate the residual at time $2n-1$, while the latter does not.

## 4.1 Vectorized SBHT RLS Array

In order to obtain the RLS filtering residual vector in the systolic array, we can use two possible approaches. The first approach is to generalize the architecture of McWhirter's Givens rotation approach [18]. A SBHT QRD array with a RA based on this approach is shown in Fig.3. Since the $\mathbf{v}_n$ in (33) results from the reflection computation in (32), therefore $\mathbf{v}_n$ is obtained naturally from the output of RA. Each boundary cell then forms the matrix $\mathbf{H}_{22}^{(\cdot)}$ and propagates it down the diagonal boundary cells. Since $\mathbf{H}_{22}^{(i)}$ is generated earlier than $\mathbf{H}_{22}^{(j)}$ for $i < j$, equation (35) has to be computed from left to right involving matrix-matrix multiplications. As a result, each boundary cell performs the matrix multiplication to cumulate $\mathbf{H}_{22}^{(\cdot)}$ when it is propagated down diagonal boundary cells. The matrix multiplications needed in the boundary cells in this approach are objectionable since they not only slow down the throughput but also increase the complexity of the boundary cells. We note, McWhirter's original approach based on Givens rotation worked well since only scalars need to be propagated down the diagonal boundary cells and the order of multiplications for the scalars is irrelevant.

Obviously, we prefer to compute (35) from right to left such that only inner product computations are performed. Instead of forming the matrix $\mathbf{H}_{22}^{(\cdot)}$ and propagating it down, another approach is to use the facts that $\mathbf{H}_{22}^{(\cdot)}$ can be expressed by using (17) and the reflection vectors are sent to the right from boundary cells as described in Section 3.1. From these observations, (35) can be computed in similar manners as performed by the internal cells. A new architecture shown in Fig.4 is thus introduced to circumvent this problem. A column array of internal cells called *backward propagation array* (BPA) is added at the right hand side to perform the *backward propagation* of $\mathbf{v}_n$. Each row, say the $i^{th}$ one, needs $2(p-i)$ delayed buffers as shown in Fig.4. The $\mathbf{v}_n$ obtained at the output of RA is then backward propagated through the BPA. From (17), each cell of this array performs the operation

$$\mathbf{H}_{22}^{(i)}(n)\tilde{\mathbf{v}}_n = \tilde{\mathbf{v}}_n - \frac{\mathbf{x}_{n,i}^{(i-1)}}{\psi_i}(\mathbf{x}_{n,i}^{(i-1)T} \cdot \tilde{\mathbf{v}}_n), \quad i = p, \cdots, 2, 1, \tag{39}$$

where $\tilde{\mathbf{v}}_n$ is an updated $\mathbf{v}_n$. This is a subset of the operations performed by the internal cell shown in (20). The residual vector is obtained from the top of the newly appended column array.

The costs for this proposed architecture are: an increased latency time from $(2p+1)t_s$ of McWhirter's Givens method to $3pt_v$, where $t_s$ represents the processing time for the scalar operations used in the Givens rotation method and $t_v$ is the processing time for vector operations used in the SBHT method; the number of delay elements needed increases from $p$ to $\sum_1^p 2(p-i) = p(p-1)$; and $p$ additional internal processing cells. The operations of the boundary and internal cells are still given in Fig.2b. These results clearly show that HT can be implemented simply on a systolic array to achieve massive parallel processing with vector operations. This provides an efficient method to obtain a high throughput rate for recursive LS filtering by using the HT method.

## 5 Two-level Pipelined Implementations

The SBHT QRD array and the RLS array discussed in the above sections are derived using the conventional Householder transformation as shown in (11). Due to the vector processing nature of the conventional method, the cells of both arrays perform vector operations such as inner products. This means the complexity of each cell is high and the I/O bandwidth is large in order to achieve an effective vector data communication. Each cell, due to the complexity of vector processing, may require a large processor. Clearly, this is not a desirable property for VLSI implementation. Thus, we are motivated to find a suitable algorithm to pipeline the data down to the word level such that the I/O bandwidth as well as the complexity can be reduced. In addition, we still wish to achieve a high throughput which is needed in many modern signal processing applications.

The conventional approach in computing Householder transformation, $\mathbf{y} = \mathbf{Tq}$, based on (11) is to first form $\mathbf{z}$ and $||\mathbf{z}||^2$ from $\mathbf{x}$ and then $\mathbf{z}^T\mathbf{q}/||\mathbf{z}||^2$ and $\mathbf{q} - 2\mathbf{z}(\mathbf{z}^T\mathbf{q}/||\mathbf{z}||^2)$ as considered before. It can be stated in the following form:
**HT Algorithm** (Conventional)

**Step 1.** $S_{xx} = ||\mathbf{x}||^2$.

**Step 2.** If $S_{xx} = 0$, then $\mathbf{y} = \mathbf{q}$.

**Step 3.** If $S_{xx} \neq 0$ then
(1) $s = \sqrt{S_{xx}}$, $\mathbf{z} = \mathbf{x} + [s, 0, 0, \cdots, 0]^T$,
(2) $\phi = S_{xx} + sx_1$, $S_{zq} = \mathbf{z}^{\mathbf{T}}\mathbf{q}$,
(3) $d = S_{zq}/\phi$, $\mathbf{y} = \mathbf{q} - d\mathbf{z}$.

In [24], Tsao pointed out that by skipping the computation of $\phi$ and avoiding the cumbersome intermediate steps of forming vector $\mathbf{z}$ for further computations, a modified algorithm with smaller round off error and less operations can be obtained. Only Step 3 of the conventional algorithm is modified as follows:
**Modified HT Algorithm** (Tsao, 1975)

**Step 3.** If $S_{xx} \neq 0$ then
(1) $s = \sqrt{S_{xx}}$, $\sigma = x_1 + s$,

(2) $S_{xq} = \mathbf{x}^{\mathbf{T}}\mathbf{q}$,
(3) $y_1 = -S_{xq}/s, \;\; d = (q_1 - y_1)/\sigma, \;\; y_i = q_i - dx_i, \; i = 2, \cdots, n.$

With this algorithm, the operations of the cells of the vectorized systolic arrays can be modified as shown in Fig.5. As we can see, for the boundary cell, the vector **u**, which consists of the weighted diagonal element of the upper-triangular matrix and one column of the input data block (updated or not), can be sent out immediately when the input **x** is available without waiting for any computations as required in the implementation using the conventional algorithm. Due to this advantage and the modified operations in the internal cells, we can then pipeline the vector operations down to the word level such that each cell only performs scalar operations which will significantly reduce the complexity of the cell.

A two-level pipelined implementation of the modified HT algorithm is given in Fig.6a. The boundary cell performs three major functions: square-and-cumulate, square root, and addition. For each data block, the boundary cell fetches one data sample, cumulates the square of the sample, and sends the data to the right for internal cell. When all the data of the block are processed, the content of $S$ is then sent down for square-root operation. The resultant $s$ is sent to the right for internal cell as well as sent down to obtain $\sigma$, which is then sent to the right when available. At the same time, when an internal cell receives a $u_i$, it multiplies $u_i$ with an input $x_i$ and cumulates all these products to obtain $S$. When $S$ is available, it is sent down for division operation with $s$, which arrives at the same time, to obtain $t$; then $t$ is sent down and $\sigma$ again arrives at the same time to compute $d$. To compute $y_i$ of (3) in Step 3, we need registers to store $u_i$ and $x_i$ temporarily. Since data from the next block is continuously being sent into the system, each internal cell needs $2(k+3)$ registers to store $u_i$ and $x_i$ as indicated in Fig.6a. When $d$ is available, $y_i$ is then obtained one by one and sent down for further processing. For data from the next block, it goes through the same processing. When a new $d$ is available in the internal cell, the corresponding $x_i$ and $u_i$ are already waiting in the registers. Therefore the vector operations are successfully pipelined down to the word level. This means that by using the modified HT algorithm, we have not only pipelined the SBHT arrays at the vector level but also at the word level. The input data is now skewed in the word level as shown in Fig.6a rather than in the vector level as shown in Fig.4. The function descriptions of the processing cells for two-level pipelined implementation are given in Fig.6b.

Since the most time consuming operation of this two-level pipelined implementation is the square root operation which is also the critical operation in McWhirter's Givens rotation implementation, the throughput of this two-level pipelined implementation is as fast as that of the McWhirter's Givens array. However, with a longer pipeline, a longer system latency for the SBHT method is obtained. This is due to the fact that the registers of the internal cells have to be all filled before we can obtain the residual vector. For the SBHT RLS systolic array of order $p$, we have $(p^2 + 3p)/2$ internal cells, including the BPA. Thus, there are a total of $(p^2 + 3p)(k+3)$ registers for the whole system. The system latency is given by $t_s = 2p(k+4)$, which is linearly proportional to $p$ and $k$. However, for the Givens rotation method, the system latency is only $t_s = 2p + 1$. Comparisons of both RLS arrays based on the SBHT and Givens rotation are summarized in Table 1. In general, the throughput of the SBHT RLS systolic array is as fast as the Givens rotation method. Of course, while the cell complexity of the SBHT array is higher, it does offer better numerical property [25]. A detailed backward error analysis carried out by Wilkinson showed that for an $n \times n$ matrix

$A$, after $n(n-1)/2$ Givens rotations, the roundoff error in the upper-triangular matrix is in the order of $\mathcal{O}(\kappa_g n^{3/2}\mu\|A\|)$ [25, pp. 138], while a series of $(n-1)$ HT gives $\mathcal{O}(\kappa_h n\mu\|A\|)$ [25, pp. 160], with $\kappa_g$ and $\kappa_h$ being constants and $\mu$ a machine floating computation precision.

## 6 Constrained RLS Problems

In the above sections, we have dealt with an unconstrained RLS problem. The RLS systolic array considered there was motivated originally by the sidelobe canceller beamformation problem [18]. Other practical motivation could have come from the adaptive filtering problem [6]. However, there are other signal processing applications which are modeled by a constrained RLS problem. The MVDR beamformation constitutes such an example [19,20,22]. It is interesting to determine whether a systolic array for an unconstrained RLS problem can also be used for a constrained RLS problem. In [19], McWhirter and Shepherd showed an extension of the unconstrained RLS array to the MVDR beamforming problem. Based on their approach, we shall also demonstrate the implementation of a MVDR beamformation problem using a SBHT RLS array.

The MVDR beamforming problem is to minimize

$$\xi^{(\ell)}(n) = \|\mathbf{X}(n)\mathbf{w}^{(\ell)}(n)\|_{\Lambda}, \quad \ell = 1,\cdots,L, \tag{40}$$

subject to the linear constraints of

$$\mathbf{c}^{(\ell)T}\mathbf{w}^{(\ell)}(n) = \beta^{(\ell)}, \quad \ell = 1,\cdots,L, \tag{41}$$

where $L$ is the number of constraints. We are interested in the *a posteriori* residual vector

$$\hat{\mathbf{e}}_n^{(\ell)}(n) = \mathbf{X}_n^T\hat{\mathbf{w}}^{(\ell)}(n). \tag{42}$$

The optimal solution of the weight vector is known [19] to be given by

$$\hat{\mathbf{w}}^{(\ell)}(n) = \frac{\beta^{(\ell)}\mathbf{M}^{-1}(n)\mathbf{c}^{(\ell)}}{\mathbf{c}^{(\ell)T}\mathbf{M}^{-1}(n)\mathbf{c}^{(\ell)}} = \frac{\beta^{(\ell)}\mathbf{R}^{-1}(n)\mathbf{a}^{(\ell)}(n)}{\|\mathbf{a}^{(\ell)}(n)\|^2}, \tag{43}$$

where $\mathbf{M} = \mathbf{X}^T(n)\Lambda_k^2(n)\mathbf{X}(n)$ is the weighted covariance matrix, $\mathbf{R}(n)$ is the upper triangular matrix resulted from the QRD of the weighted data matrix $\Lambda_k\mathbf{X}(n)$, and

$$\mathbf{a}^{(\ell)}(n) = \mathbf{R}^{-T}(n)\mathbf{c}^{(\ell)}. \tag{44}$$

Therefore the optimal residual vector at time $n$ is

$$\hat{\mathbf{e}}_n^{(\ell)}(n) = \frac{\beta^{(\ell)}}{\|\mathbf{a}^{(\ell)}(n)\|^2}\cdot\mathbf{X}_n^T\mathbf{R}^{-1}(n)\mathbf{a}^{(\ell)}(n). \tag{45}$$

A crucial step needed is for the efficient recursive updating of $\mathbf{a}^{(\ell)}(n)$. A novel approach was proposed for performing this updating [19]. Specifically, from (8), (9), and (44),

$$\begin{aligned}\mathbf{c}^{(\ell)} &= \mathbf{R}^T(n-1)\mathbf{a}^{(\ell)}(n-1)\\ &= \lambda^{-2}\left[\begin{array}{ccccc}\lambda\mathbf{R}^T(n-1) & \vdots & \mathbf{0}^T & \vdots & \mathbf{X}_n\end{array}\right]\left[\begin{array}{c}\lambda\mathbf{a}^{(\ell)}(n-1)\\ ----\\ \lambda\mathbf{b}^{(\ell)}(n-1)\\ ----\\ \mathbf{0}\end{array}\right],\end{aligned} \tag{46}$$

where $\mathbf{b}^{(\ell)}(n-1)$ is an arbitrary $((n-1)k-p)\times 1$ vector. Then from *Lemma 1*, (8), and (9), we have

$$\begin{aligned}
\mathbf{c}^{(\ell)} &= \lambda^{-2}\left[\ \lambda\mathbf{R}^T(n-1)\ \vdots\ \mathbf{0}^T\ \vdots\ \mathbf{X}_n\ \right]\mathbf{H}^T(n)\mathbf{H}(n)\begin{bmatrix}\lambda\mathbf{a}^{(\ell)}(n-1)\\ ----\\ \lambda\mathbf{b}^{(\ell)}(n-1)\\ ----\\ \mathbf{0}\end{bmatrix}\\
&= \mathbf{R}^T(n)\cdot\lambda^{-2}(\lambda\mathbf{H}_{11}(n)\mathbf{a}^{(\ell)}(n-1)). \qquad (47)
\end{aligned}$$

Thus, $\mathbf{a}^{(\ell)}(n)=\lambda^{-2}(\lambda\mathbf{H}_{11}(n)\mathbf{a}^{(\ell)}(n-1))$ can be obtained by updating $\mathbf{a}^{(\ell)}(n-1)$ in a way similar to that $\mathbf{u}(n)$ is obtained by updating $\mathbf{u}(n-1)$ using (32). The only differences are the input for updating $\mathbf{a}^{(\ell)}(n-1)$ is a zero vector and a scaling factor $\lambda^{-2}$. Due to the structure of $\mathbf{H}$ in *Lemma 1*, the vector $\mathbf{b}^{(\ell)}(\cdot)$ plays no role in the updating of $\mathbf{a}^{(\ell)}(\cdot)$. Furthermore, from (27) and (29), we have

$$\hat{\mathbf{e}}_n(n)=\mathbf{X}_n^T\mathbf{R}^{-1}(n)\mathbf{u}(n)-\mathbf{y}_n. \qquad (48)$$

From (32), we see that $\mathbf{u}(n)$ results from the update of $[\mathbf{y}(n-1)\vdots\mathbf{y}_n]$, where $\mathbf{y}_n$ is the new input. Now replacing $\mathbf{u}(n)$ with $\mathbf{a}^{(\ell)}(n)$ and $\mathbf{y}_n$ with a zero vector, we have

$$\hat{\mathbf{e}}_n(n)=\mathbf{X}_n^T\mathbf{R}^{-1}(n)\mathbf{a}^{(\ell)}(n), \qquad (49)$$

and from (45), we then obtain

$$\hat{\mathbf{e}}_n^{(\ell)}(n)=\frac{\beta^{(\ell)}}{||\mathbf{a}^{(\ell)}(n)||^2}\cdot\hat{\mathbf{e}}_n(n). \qquad (50)$$

This equation reveals that by the proper scaling of $\hat{\mathbf{e}}_n(n)$, which can be obtained from the SBHT RLS systolic array, we can obtain the *a posteriori* residual vector, $\hat{\mathbf{e}}_n^{(\ell)}(n)$, of the MVDR beamformation. Fig.7 shows an extension of the SBHT RLS array for the new problem. Now one more data channel is needed for the RA to pipeline cumulation of $||\mathbf{a}^{(\ell)}(n)||^2$, and the scaling of the residual vector is done at the bottom of the RA when a new $||\mathbf{a}^{(\ell)}(\cdot)||^2$ is available. Each RA/BPA pair in Fig.7 represents one of the $K$ constraints. The optimal *a posteriori* residual vector of each linear constraint is obtained at the output of the corresponding backward propagation array.

As pointed out in [19], there are two ways to initialize the array. One method is to set $\mathbf{R}(0)=\delta\mathbf{I}$, where $\delta$ is a small scalar, and thus from (44), $\mathbf{a}^{(\ell)}(0)=\delta^{-1}\mathbf{c}^{(\ell)}$, $\ell=1,\cdots,L$. Another method is to obtain $\mathbf{R}(n)$ to some time $n$, then use (44) to obtain $\mathbf{a}^{(\ell)}(n)$. The details of a two-mode operation required for this initialization procedure are also considered in [19].

## 7 Conclusions

In this paper, we have shown that the Householder transformation can be implemented on a systolic array. By using a two-level pipelined implementation, the throughput of the

SBHT RLS systolic array can be as fast as that of the original Givens array in [18]. While, the system latency is longer for the SBHT, it provides a better numerical stability than the Givens method. Clearly, the Givens array is a special case of the SBHT array with a block size of one. In general, the block size is an important variable. A larger block size results in a better numerical stability, while the system latency is increased. Many known properties of the Givens array are also applicable to the SBHT array. For example, the real-time algorithm-based fault-tolerant scheme proposed in [14] can also be easily incorporated into the SBHT RLS array. From the results described in this paper, it appears that the Householder transformation method is useful in real-time high throughput applications of modern signal processing as well as in VLSI implementation.

## References

[1] M.G. Bellanger, "Computational complexity and accuracy issues in fast least squares algorithms for adaptive filtering", Proc. IEEE ISCAS, pp.2635-2639, Finland, 1988.

[2] J.M. Cioffi, "Limited-precision effects in adaptive filtering", IEEE Trans. CAS, VOL CAS-34, pp.821-833, July, 1987.

[3] J.M. Cioffi, "The fast Householder filters RLS adaptive filter", Proc. IEEE ICASSP, pp.1619-1622, Albuquerque, April 1990.

[4] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays", Proc. SPIE, Vol 298, Real Time Signal Processing IV, pp.298, 1981.

[5] G. H. Golub and C. F. Van Loan, **Matrix Computations,** 2nd ed., Baltimore, MD: Johns Hopkins University Press, 1989.

[6] S. Haykin, **Adaptive Filter Theory**, Prentice Hall, 1986.

[7] D.E. Heller and I.C.F. Ipsen, "Systolic networks for orthogonal decomposition", SIAM J. Sci. Stat. Comput. Vol 4, pp.261-269, June, 1983.

[8] L. Johnsson, "A computational array for the QR-method," 1982 Conference on Advanced Research in VLSI, M.I.T., pp. 123-129.

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

[10] H.T. Kung and M.S. Lam, "Wafer-scale integration and two-level pipelined implementation of systolic array", J. Parallel Distrib. Comput., Vol 1, pp.32-63, 1984.

[11] S.Y. Kung, **VLSI Array Processors**, Prentice Hall, 1988.

[12] H. Leung and S. Haykin, "Stability of recursive QRD LS algorithms using finite-precision systolic array implementation", IEEE Trans. ASSP, VOL37 pp.760-763, May 1989.

[13] F. Ling, D. Manolakis, and J.G. Proakis, "A recursive modified Gram-Schmidt algorithm for least-squares estimation", IEEE Trans. ASSP, Vol. ASSP-34, pp.829-836, Aug. 1986.

[14] K.J.R. Liu and K. Yao, "Gracefully degradable real-time algorithm-based fault-tolerant method for QR recursive least-squares systolic array", Proc. International Conference on Systolic Array, pp. 401-410, Killarney, Ireland, May, 1989.

[15] K.J.R. Liu, S.F. Hsieh, and K. Yao, "Recursive LS filtering using block Householder transformations", Proc. IEEE ICASSP, pp.1631-1634, Albuquerque, April 1990.

[16] F.T. Luk, "A rotation method for computing the QR-decomposition", SIAM J. Sci. Stat. Comput. Vol 7, pp.452-459, Apr. 1986.

[17] F. T. Luk and S. Qiao, "Analysis of a recursive least-squares signal-processing algorithm," SIAM J. Sci. Stat. Comput. Vol. 10, No. 3, pp. 407-418, May 1989.

[18] J. G. McWhirter, "Recursive least-squares minimisation using a systolic array," Proc. SPIE, Vol. 431, Real Time Signal Processing VI, pp. 105-112, 1983.

[19] J. G. McWhirter and T. J. Shepherd, "Systolic array processor for MVDR beamforming," IEE Proceedings, Vol. 136, Pt. F, No. 2, pp. 75-80, 1989.

[20] N.L. Owsley, "Sonar array processing", in **Array Signal Processing**, pp.115-193, Haykin, Ed., Prentice-Hall, 1985.

[21] B. N. Parlett, "Analysis of algorithms for reflections in bisectors," SIAM Review, Vol. 13, No. 2, pp. 197-208, Apr. 1971.

[22] R. Schreiber, "Implementation of adaptive array algorithms", IEEE Trans. ASSP Vol ASSP-34, pp.1038-1045, Oct. 1986.

[23] A. Steinhardt, "Householder transformation," IEEE ASSP Mag., 1988.

[24] N.-K. Tsao, "A note on implementing the Householder transformation," *SIAM J. Numer. Anal.*, Vol. 12, No. 1, pp. 53-58, Mar. 1975.

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

[26] J. H. Wilkinson, "Modern error analysis," SIAM Review, Vol. 13, No. 4, pp. 548-568, Oct. 1971.

**Figure Captions:**

**Fig.1a** QRD RLS systolic array using Givens rotation method.

**Fig.1b** Processing cells of the Givens rotation method.

**Fig.2a** SBHT QRD systolic array.

**Fig.2b** Processing cells of the SBHT QRD systolic array.

**Fig.3** SBHT RLS systolic array obtained by direct generalization of the Givens rotation array.

**Fig.4** New matrix multiplication free SBHT RLS systolic array.

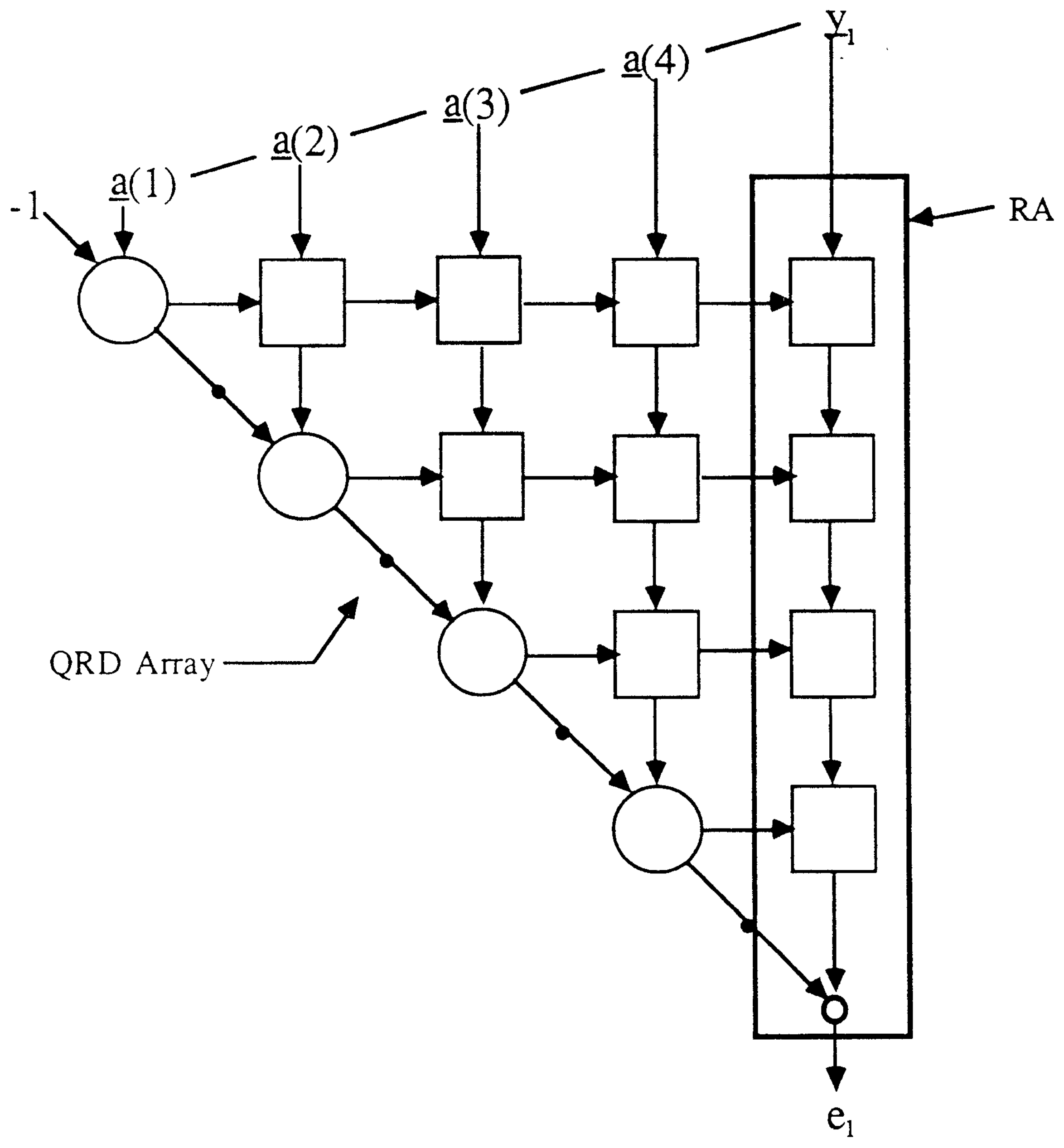
**Fig.5** Operations of the processing cells by using modified Householder transformation.

**Fig.6a** A architectures of the processing cells for two-level pipelined implementation.

**Fig.6b** Functional descriptions of processing cells for two-level pipelined implementation.

**Fig.7** MVDR beamforming systolic array.

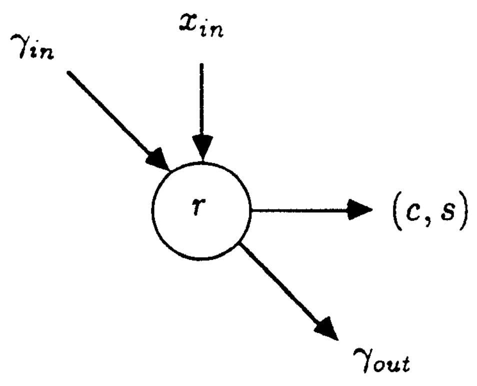
**Table 1** Comparisons of the SBHT and Givens rotation methods.



$\underline{a}(1)$
$\underline{a}(2)$
$\underline{a}(3)$
$\underline{a}(4)$
$y_1$
-1
RA
QRD Array
$e_1$

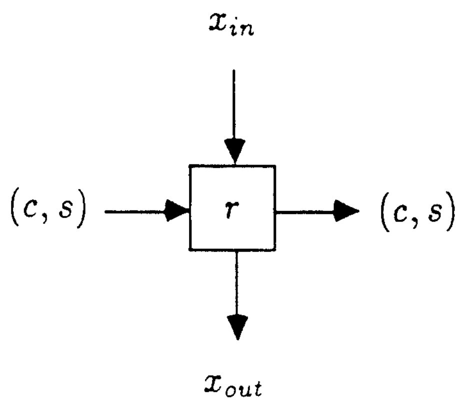
Fig. 1a

(1) Boundary Cell



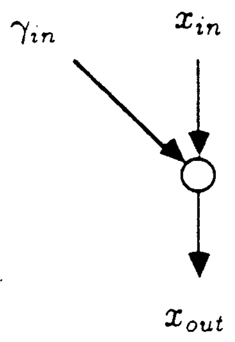
```
If $x_{in} = 0$ then
    $c \leftarrow 1;\ s \leftarrow 0;\ \gamma_{out} \leftarrow \gamma_{in};$
    $r = \lambda r,$
otherwise
    $r' = \sqrt{\lambda^2 r^2 + x_{in}^2};$
    $c \leftarrow \lambda r / r';\ s \leftarrow x_{in} / r'$
    $r \leftarrow r';\ \gamma_{out} = c\gamma_{in}$
end
```

(2) Internal Cell



$$x_{out} \leftarrow cx_{in} - s\lambda r$$
$$r \leftarrow sx_{in} + c\lambda r$$

(3) Final Cell



$$x_{out} \leftarrow \gamma_{in} x_{in}$$

Fig. 1b

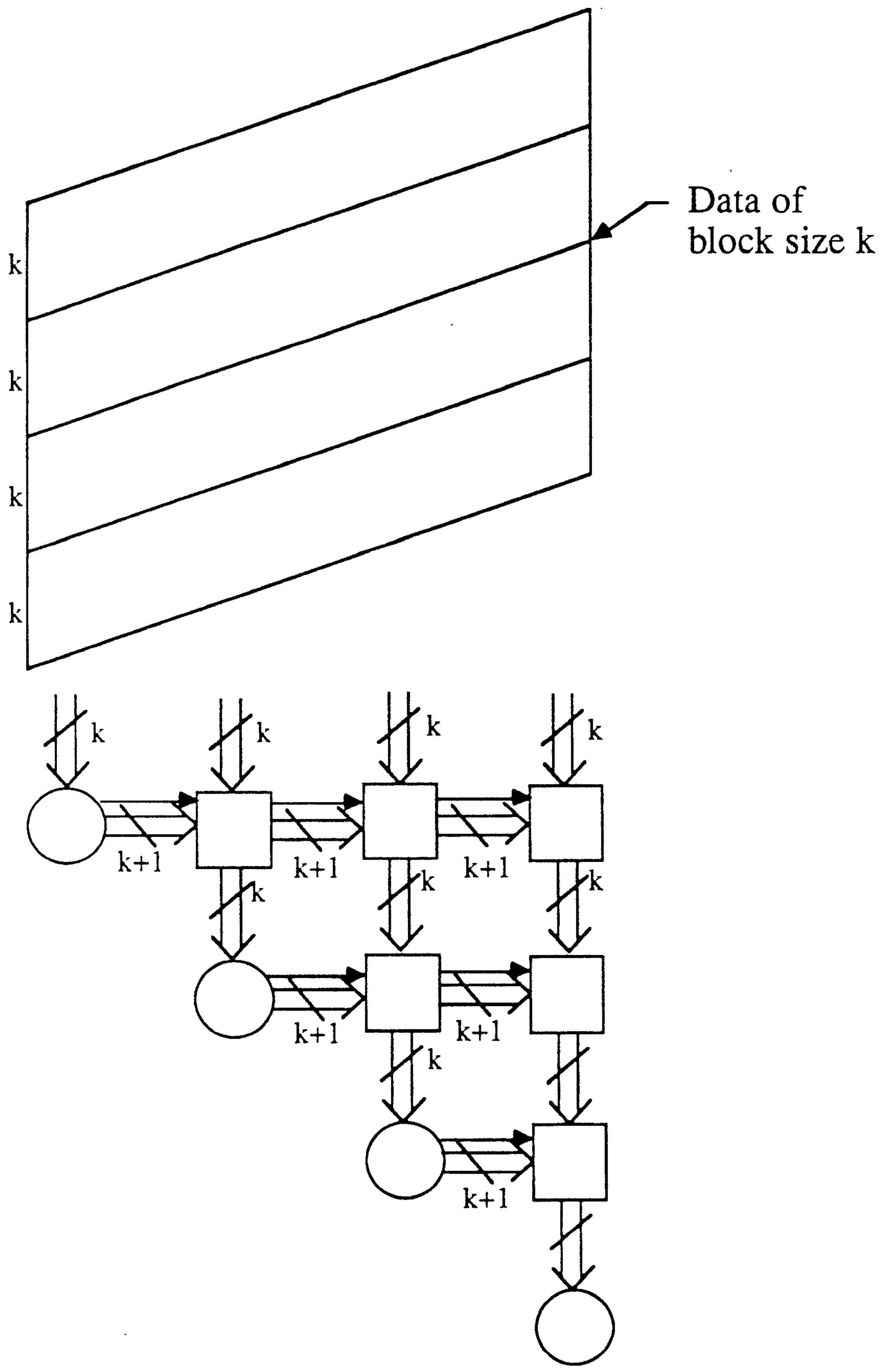
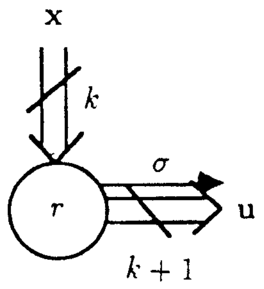
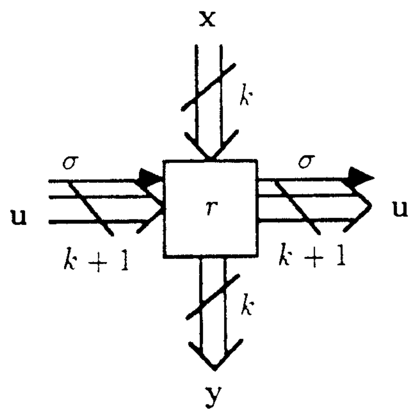


Fig.2a



```
If $\|\mathbf{x}\|^2 = 0$ then
    $\sigma = 0,\ r = \lambda r,$
else
    $S \leftarrow \lambda^2 r^2 + \|\mathbf{x}\|_2^2$
    $s \leftarrow \sqrt{S}$
    $\sigma \leftarrow S + s\lambda r$
    $\mathbf{u} \leftarrow \begin{bmatrix} \lambda r + s \\ \mathbf{x} \end{bmatrix}$
    $r \leftarrow s$
end
```



```
If $\sigma = 0$ then
    $\mathbf{y} = \mathbf{x},\ r = \lambda r,$
else
    $t \leftarrow \sigma^{-1} \cdot \mathbf{u}^T \cdot \begin{bmatrix} \lambda r \\ \mathbf{x} \end{bmatrix}$
    $\begin{bmatrix} r \\ \mathbf{y} \end{bmatrix} \leftarrow \begin{bmatrix} \lambda r \\ \mathbf{x} \end{bmatrix} - t \cdot \mathbf{u}$
end
```

Fig.2b

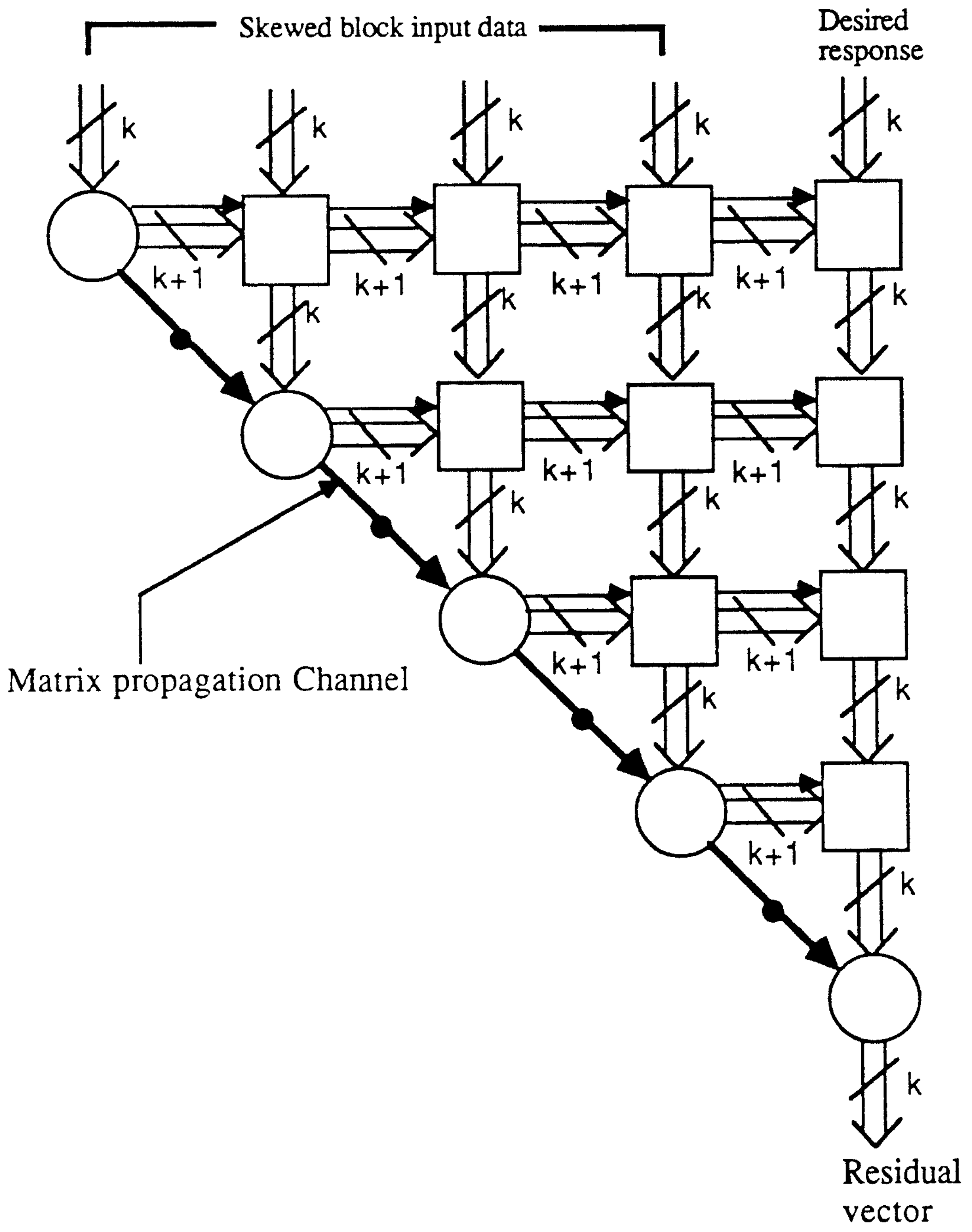

Skewed block input data
Desired response
k
k+1
Matrix propagation Channel
Residual vector

Fig.3

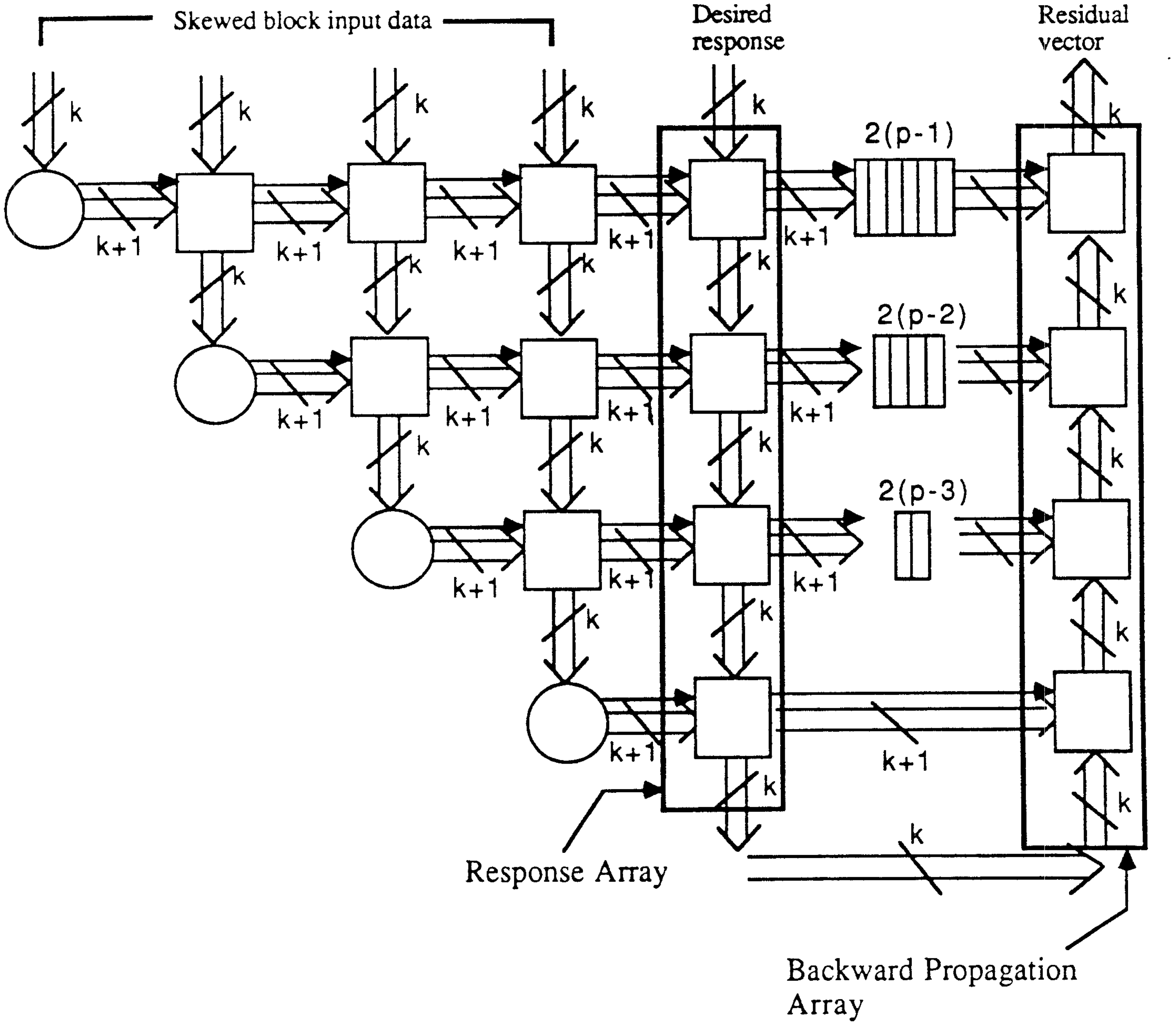
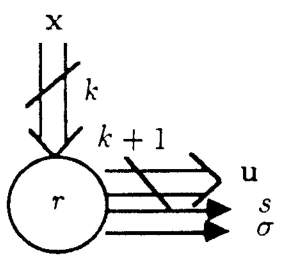
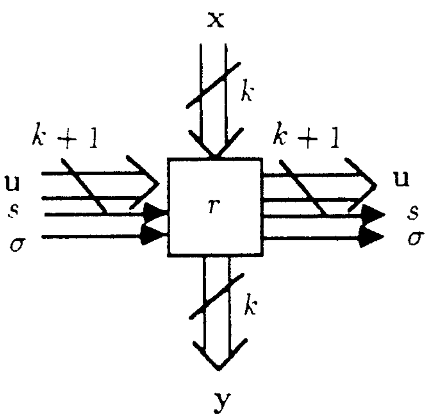

Skewed block input data
Desired response
Residual vector
k
k+1
2(p-1)
2(p-2)
2(p-3)
Response Array
Backward Propagation Array

Fig.4



```
If $\|\mathbf{x}\|^2 = 0$ then
    $\sigma = 0,\ r = \lambda r,$
else
    $S \leftarrow \lambda^2 r^2 + \|\mathbf{x}\|_2^2$
    $s \leftarrow \sqrt{S}$
    $\sigma \leftarrow s + \lambda r$
    $\mathbf{u} \leftarrow \begin{bmatrix} \lambda r \\ \mathbf{x} \end{bmatrix}$
    $r \leftarrow s$
end
```



```
If $\sigma = 0$ then
    $\mathbf{y} = \mathbf{x},\ r = \lambda r,$
else
    $S \leftarrow \mathbf{u}^T \cdot \begin{bmatrix} \lambda r \\ \mathbf{x} \end{bmatrix}$
    $t \leftarrow -S/s$
    $d \leftarrow (\lambda r - t)/\sigma$
    $r \leftarrow t$
    $\mathbf{y} \leftarrow \mathbf{x} - d \cdot \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_{k+1} \end{bmatrix}$
end
```

Fig.5

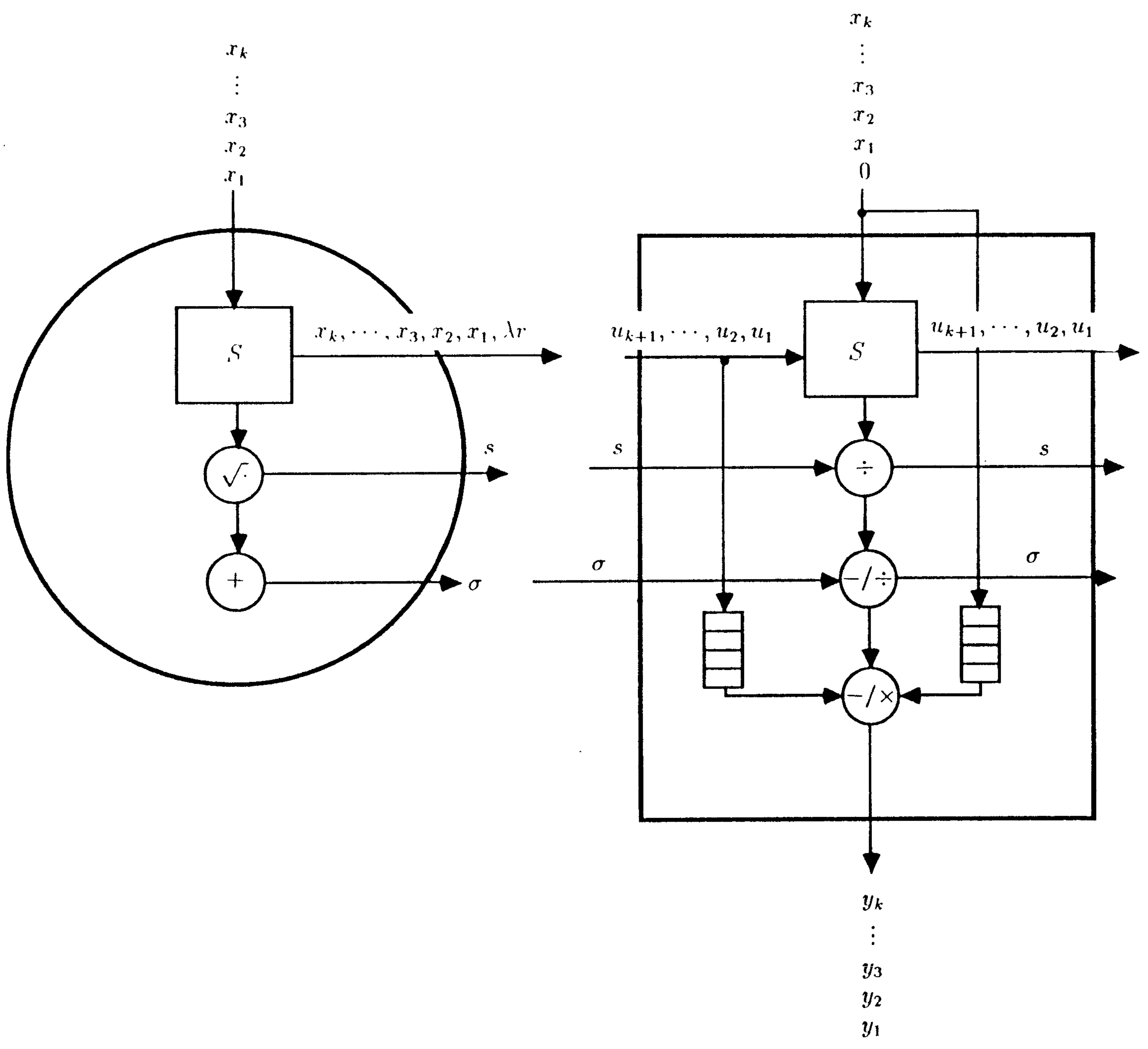
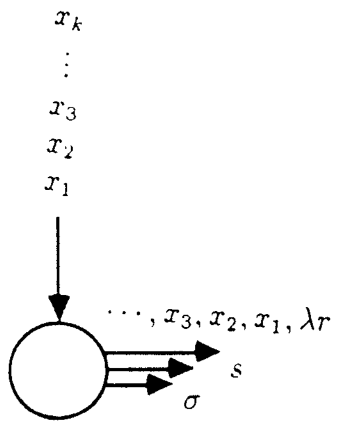
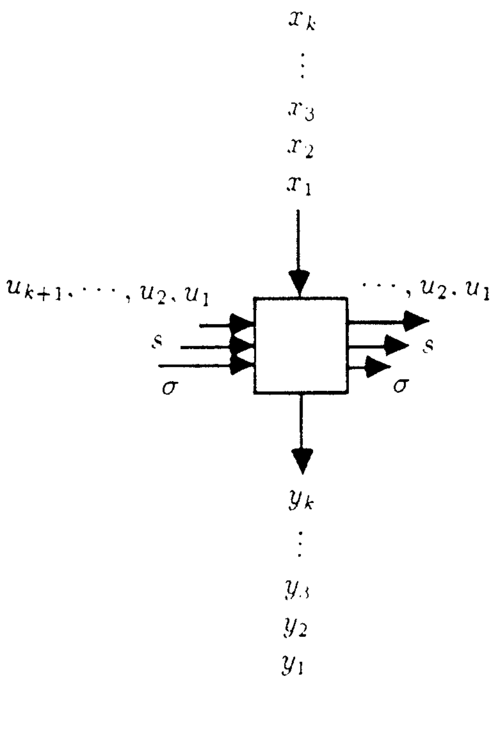

$x_k$
$\vdots$
$x_3$
$x_2$
$x_1$
$S$
$x_k, \cdots, x_3, x_2, x_1, \lambda r$
$\sqrt{\cdot}$
$s$
$+$
$\sigma$
$x_k$
$\vdots$
$x_3$
$x_2$
$x_1$
$0$
$u_{k+1}, \cdots, u_2, u_1$
$S$
$u_{k+1}, \cdots, u_2, u_1$
$s$
$\div$
$s$
$\sigma$
$-/\div$
$\sigma$
$-/\times$
$y_k$
$\vdots$
$y_3$
$y_2$
$y_1$

Fig.6a

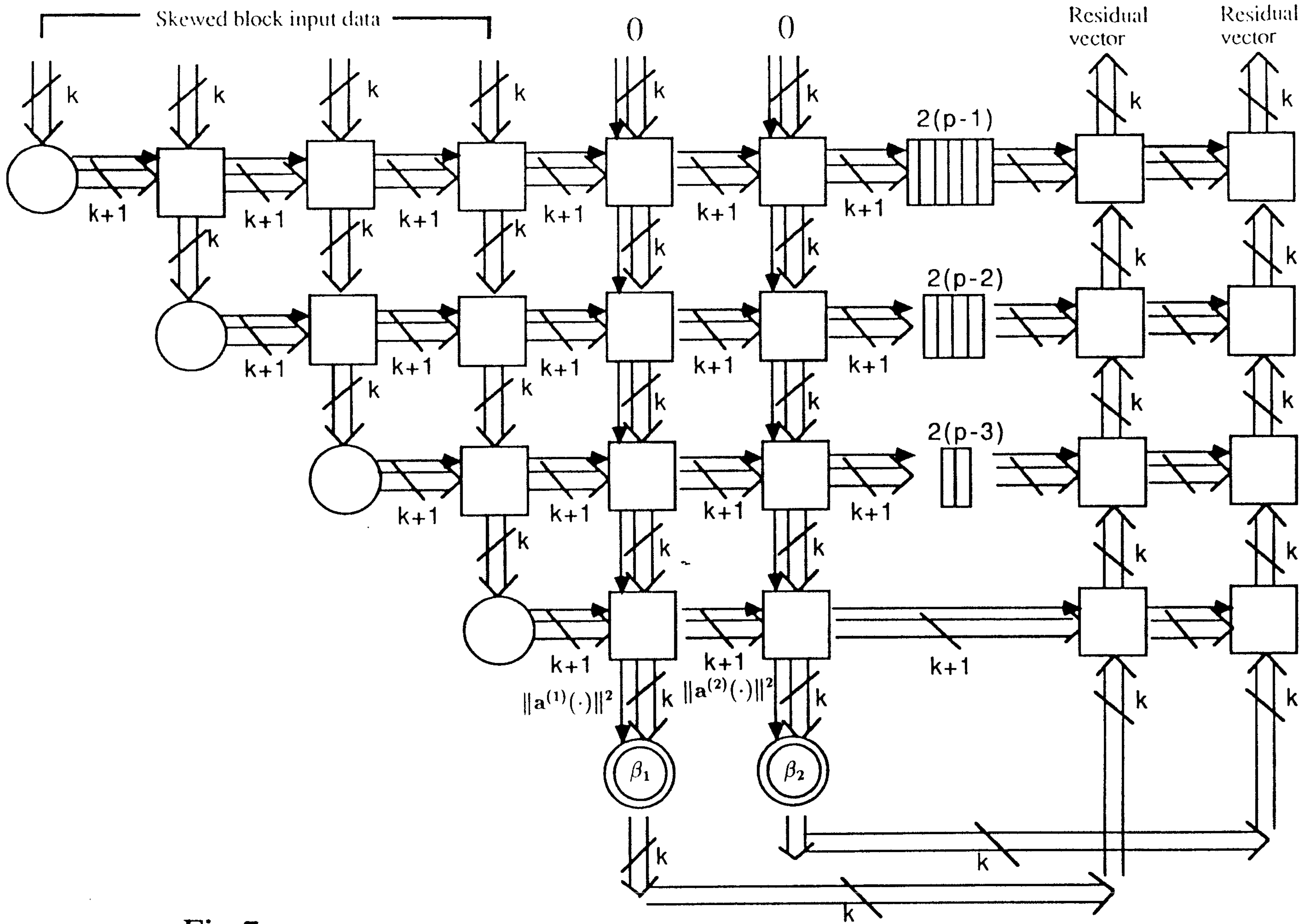


```
send_right $\lambda r$
$S \leftarrow \lambda^2 r^2$
do $i = 1, k$
   fetch $x_i$; $S \leftarrow S + x_i^2$; send_right $x_i$,
end do
$r' \leftarrow \lambda r$; $s \leftarrow \sqrt{S}$; send_right $s$,
$r \leftarrow s$; $\sigma \leftarrow r' + s$; send_right $\sigma$.
```



```
$S = 0$.
do $i = 1, K + 1$
fetch $u_i$,
if $i = 1$ then
$S \leftarrow S + u_i \lambda r$; send_right $u_i$,
else
fetch $x_{i-1}$; $S = S + u_i x_{i-1}$; send_right $u_i$.
end if
fetch $s$; $r' \leftarrow \lambda r$; $t \leftarrow -S/s$,
fetch $\sigma$; $r \leftarrow t$; $d \leftarrow (r' - t)/\sigma$,
end do
do $i = 1, k$
$y_i = x_i - d u_{i+1}$; send_down $y_i$.
end do
```

Fig.6b



Skewed block input data
0
0
Residual vector
Residual vector
k
k+1
2(p-1)
2(p-2)
2(p-3)
$\|a^{(1)}(\cdot)\|^2$
$\|a^{(2)}(\cdot)\|^2$
$\beta_1$
$\beta_2$

Fig.7

|  | Givens rotation | SBHT |
|---|---|---|
| Number of cells | $(p^2+3p)/2$ | $(p^2+5p)/2$ |
| Number of delay elements | $p$ | $p(p-1)$ |
| Number of registers | 0 | $(p^2+3p)(k+3)$ |
| System latency | $2p+1$ | $2p(k+4)$ |
| Cell complexity | less | higher |
| Numerical stability | good | better |

Table 1