D10b.4

Recursive LS Filtering Using Block Householder Transformations

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

Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90024-1594

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 method has been successfully applied to the development of the QRD RLS systolic array. It is well-known that Householder transformations (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 Transformations*, 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. Our approach makes the HT amendable for VLSI implementation as well as applicable to real-time high throughput applications of modern signal processing.

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. [4]. 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, 4]. Unfortunately, these methods can encounter numerical difficulties due to the acculumation of round-off errors under a finite-precision implementation. This may lead to a divergence of the computations of the RLS algorithm. A new type of algorithm based on the QR decomposition (QRD) known as the QRD RLS was first proposed by McWhirter in [8]. This algorithm is one of the most promising algorithms in that it is numerical stable [1] as well as suitable for parallel processing implementation on a systolic array [4, 8].

Up to now, most of the QRD RLS implementations were based on the Givens rotation method which is a rank-1 update approach [2, 7, 8]. It is well-known that the Householder transformations (HT), which is a rank-k update approach, is one of the most computationally efficient method to compute QRD. The error analysis carried out by Wilkinson [9, 5] showed that HT outperforms Givens method under finite precision computations.

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

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 transformations. 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 amendable 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 [2] and McWhirter's QRD RLS systolic array [8] respectively.

2 Systolic Block Householder Transformations

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

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

and the *desired response vector* as

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

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

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

and $\mathbf{y}_i$ is the i^{th} desired response block,

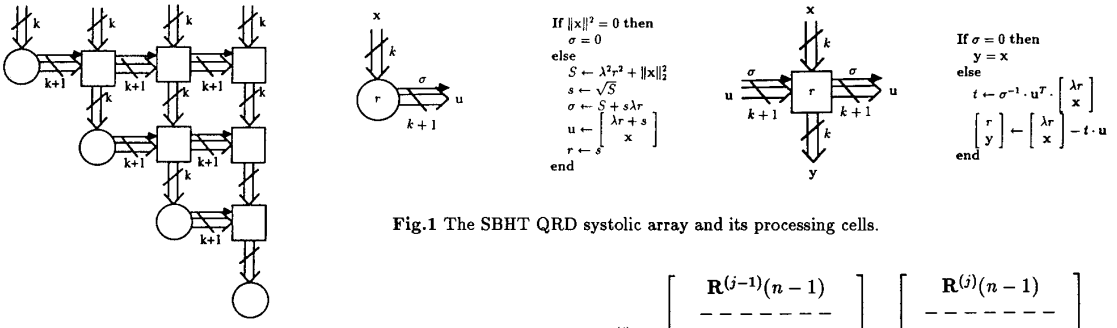

Fig.1 The SBHT QRD systolic array and its processing cells.

$$\mathbf{y}_i = \begin{bmatrix} y_{(i-1)k+1} \\ y_{(i-1)k+2} \\ \vdots \\ y_{ik} \end{bmatrix} \in \Re^k. \tag{4}$$

k is the *block size* and p is the *order* (columns) of the system.

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

$$\mathbf{Q}(n-1)\mathbf{X}(n-1) = \begin{bmatrix} \mathbf{R}(n-1) \\ --- \\ 0 \end{bmatrix}. \tag{5}$$

Denote

$$\bar{\mathbf{Q}}_k(n-1) = \begin{bmatrix} \mathbf{Q}(n-1) & \vdots & 0 \\ --- & & --- \\ 0 & \vdots & \mathbf{I}_k \end{bmatrix}, \tag{6}$$

then we have

$$\bar{\mathbf{Q}}_k(n-1)\mathbf{X}(n) = \begin{bmatrix} \mathbf{R}(n-1) \\ --- \\ 0 \\ --- \\ \mathbf{X}_n^T \end{bmatrix}. \tag{7}$$

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

$$\mathbf{H}(n)\bar{\mathbf{Q}}_k(n-1)\mathbf{X}(n) = \begin{bmatrix} \mathbf{R}(n) \\ --- \\ 0 \end{bmatrix}, \tag{8}$$

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

$$\mathbf{Q}(n) = \mathbf{H}(n)\bar{\mathbf{Q}}_k(n-1). \tag{9}$$

A $n \times n$ Householder transformations matrix $\mathbf{T}$ is of the form

$$\mathbf{T} = \mathbf{I} - \frac{2\mathbf{z}\mathbf{z}^{\mathrm{T}}}{\|\mathbf{z}\|^2}, \tag{10}$$

where $\mathbf{z} \in \Re^n$. 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

$$\mathbf{T}\mathbf{x} = \pm\|\mathbf{x}\|_2\mathbf{e}_1. \tag{11}$$

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,

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

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

$$\mathbf{H}(n) = \mathbf{H}^{(p)}(n)\mathbf{H}^{(p-1)}(n)\cdots\mathbf{H}^{(1)}(n), \tag{12}$$

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

In general,

$$\mathbf{H}^{(m)} = \begin{bmatrix} \mathbf{H}_{11}^{(m)} & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{12}^{(m)} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{0} & \vdots & \mathbf{I}_{(n-1)k-p} & \vdots & \mathbf{0} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{H}_{21}^{(m)} & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{22}^{(m)} \end{bmatrix}, \tag{13}$$

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

$$\mathbf{H}_{22}^{(m)} = \mathbf{I} - \frac{2\mathbf{x}_{n,m}^{(m-1)}\mathbf{x}_{n,m}^{(m-1)T}}{\sigma_{z_m}^2} \in \Re^{k \times k} \tag{14}$$

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

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

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{12} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{0} & \vdots & \mathbf{I}_{(n-1)k-p} & \vdots & \mathbf{0} \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ \mathbf{H}_{21} & \vdots & \mathbf{0} & \vdots & \mathbf{H}_{22} \end{bmatrix} = \mathbf{H}^{(p)}\mathbf{H}^{(p-1)}\cdots\mathbf{H}^{(1)}, \tag{15}$$

with

$$\mathbf{H}_{ij} = \mathbf{H}_{ij}^{(p)}\cdots\mathbf{H}_{ij}^{(2)}\mathbf{H}_{ij}^{(1)} \tag{16}$$

If the block size $k = 1$, due to the fact that the Givens rotation method is a special case of rank-1 update Householder transformations [3], the $\mathbf{H}$ matrix in *Lemma 1* becomes a Givens rotation matrix $\mathbf{G}$ of the form as given in [8].

2.1 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 [2], this array has both boundary and internal cells. The boundary cell takes an input of block size k from the previous 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 processings. Define

$$\bar{\mathbf{x}}_{n,i}^{(i)} = \left[\begin{array}{ccccc} \mathbf{0}_{i-1} & \vdots & r_{ii} & \vdots & \mathbf{0}_{(n-1)k-i} \end{array} \vdots \ \mathbf{x}_{n,i}^{(i-1)} \right], \quad i = 1, \cdots, p.$$

When an internal cell receives the reflection vector, instead of forming the matrix $\mathbf{z}_i\mathbf{z}_i^T$ and performing matrix operations, it performs an inner product operation to update its content by doing

$$\mathbf{H}^{(i)}\bar{\mathbf{x}}_{n,i}^{(i)} = \bar{\mathbf{x}}_{n,i}^{(i)} \frac{2\mathbf{z}_i}{\sigma_{\mathbf{z}_i}^2}(\mathbf{z}_i^T \cdot \bar{\mathbf{x}}_{n,i}^{(i)}), \tag{17}$$

and sends the reflected data downward for further processing. Fig.1 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.

3 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

$$\underline{\varepsilon}(n) = \begin{bmatrix} \mathbf{e}_1(n) \\ \mathbf{e}_2(n) \\ \vdots \\ \mathbf{e}_n(n) \end{bmatrix} = \mathbf{X}(n)\mathbf{w}(n) - \mathbf{y}(n) \tag{18}$$

is minimized. That is,

$$\hat{\mathbf{w}}(n) = \arg\min_{w} \|\underline{\varepsilon}(n)\|_{\Lambda_k}^2, \tag{19}$$

where

$$\|\underline{\varepsilon}(n)\|_{\Lambda_k} = \|\mathbf{\Lambda}_k(n)\underline{\varepsilon}(n)\|^2 = \sqrt{\sum_{i=1}^{n} \lambda^{2(n-i)} \cdot \|\mathbf{e}_i(n)\|_2^2}, \tag{20}$$

where $0 < \lambda \leq 1$ and $\mathbf{\Lambda}_k(n)$ is a block-diagonal exponential weighting matrix of the form,

$$\mathbf{\Lambda}_k(n) = \begin{bmatrix} \lambda^{n-1}\mathbf{I}_k & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \lambda\mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I}_k \end{bmatrix} \in \Re^{nk \times nk} \tag{21}$$

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

$$\|\mathbf{e}_i(n)\|_2^2 = \sum_{j=1}^{k} |e_{(i-1)k+j}(n)|^2. \tag{22}$$

The exponential forgetting weightings are incorporated in the RLS filtering scheme to avoid overflow in the processors as well as to facilitate nonstationary data updating.

If we know the QRD of the *weighted augmented data matrix* at time n (in the block sense, which is equivalent to the nk snapshots), which is given by

$$\mathbf{\Lambda}_k(n)\left[\mathbf{X}(n) \vdots \mathbf{y}(n)\right] = [\mathbf{Q}_1^T(n) \vdots \mathbf{Q}_2^T(n)] \begin{bmatrix} \mathbf{R}(n) & \vdots & \mathbf{u}(n) \\ \mathbf{0} & \vdots & \mathbf{v}(n) \end{bmatrix}, \tag{23}$$

where $\mathbf{Q}_1(n) \in \Re^{p \times nk}$ and $\mathbf{Q}_2(n) \in \Re^{(nk-p) \times nk}$ constitute an orthogonal transformation matrix with the former spanning the column space of the weighted data matrix $\mathbf{\Lambda}_k(n)\mathbf{X}(n)$ and the latter the null space, and $\mathbf{R}(n) \in \Re^{p \times p}$ is an upper triangular matrix. The optimal weight vector can be obtained by solving

$$\mathbf{R}(n)\hat{\mathbf{w}}(n) = \mathbf{u}(n). \tag{24}$$

Obviously, $\mathbf{\Lambda}_k(n)\mathbf{X}(n) = \mathbf{Q}_1^T(n)\mathbf{R}(n)$. As a result, the weighted optimal residual of (18) is,

$$\mathbf{\Lambda}_k(n)\underline{\hat{\varepsilon}}(n) = -\mathbf{Q}_2^T(n)\mathbf{v}(n), \tag{25}$$

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,

$$\hat{\mathbf{e}}_n(n) = \mathbf{X}_n^T\hat{\mathbf{w}}(n) - \mathbf{y}_n. \tag{26}$$

By recursion on n, we relate $\mathbf{Q}(n)$ and $\mathbf{Q}(n-1)$ using (9) and have

$$\mathbf{Q}(n) = \begin{bmatrix} \mathbf{H}_{11}(n)\mathbf{Q}_1(n-1) & \vdots & \mathbf{H}_{12}(n) \\ \hline \mathbf{Q}_2(n-1) & \vdots & \mathbf{0} \\ \hline \mathbf{H}_{21}(n)\mathbf{Q}_1(n-1) & \vdots & \mathbf{H}_{22} \end{bmatrix}. \tag{27}$$

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

$$\begin{bmatrix} \mathbf{u}(n) \\ \hline \mathbf{v}(n) \end{bmatrix} = \begin{bmatrix} \lambda\mathbf{H}_{11}(n)\mathbf{u}(n-1) + \mathbf{H}_{12}(n)\mathbf{y}_n \\ \hline \lambda\mathbf{v}(n-1) \\ \hline \mathbf{v}_n \end{bmatrix}, \tag{28}$$

where

$$\mathbf{v}_n = \lambda\mathbf{H}_{21}(n)\mathbf{u}(n-1) + \mathbf{H}_{22}(n)\mathbf{y}_n. \tag{29}$$

Therefore from (25), the weighted optimal residual vector can be obtained from parameters in time $n-1$ by

$$\mathbf{\Lambda}_k(n)\underline{\hat{\varepsilon}}(n) = \begin{bmatrix} -\lambda\mathbf{Q}_2^T(n-1)\mathbf{v}(n-1) - \mathbf{Q}_1^T(n-1)\mathbf{H}_{12}^T(n)\mathbf{v}_n \\ \hline -\mathbf{H}_{22}^T(n)\mathbf{v}_n \end{bmatrix}. \tag{30}$$

The new n^{th} block of the optimal residual is then obtained as

$$\hat{\mathbf{e}}_n(n) = -\mathbf{H}_{22}^T(n)\mathbf{v}_n = -\mathbf{H}_{22}^{(1)}(n)\mathbf{H}_{22}^{(2)}(n)\cdots\mathbf{H}_{22}^{(p)}(n)\mathbf{v}_n. \tag{31}$$

Note that there is difference in the optimal residuals estimated by SBHT method and Givens rotation method. To be specific, the optimal residual vector in (31) is given by

$$\hat{\mathbf{e}}_n(n) = \begin{bmatrix} e_{(n-1)k}((n-1)k+1|nk) \\ \vdots \\ e_{nk-2}(nk-1|nk) \\ e_{nk-1}(nk|nk) \end{bmatrix}, \tag{32}$$

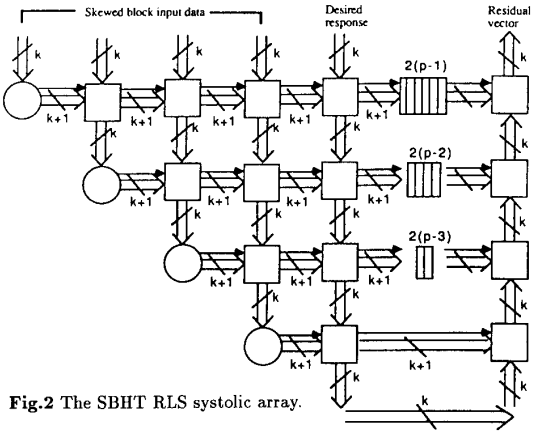

Fig.2 The SBHT RLS systolic array.

where $f(m|n)$ denotes the estimation of f on time m given the data up to time n. While the optimal residual estimated by the Givens rotation method is

$$\hat{e}_n(n) = e_n(n|n). \tag{33}$$

In this sense, the SBHT RLS gives a better estimate of the residual since it uses more data samples to estimate the optimal residual.

3.1 Vectorized SBHT RLS Array

To obtain the residual vector for RLS filtering in the systolic array, there are two possible approaches. The first one is to generalize the architecture of McWhirter's Givens rotation approach [8]. Observe that $\mathbf{v}_n$ in (29) results from the reflection computation in (28), 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 diagonal boundary cells. Since $\mathbf{H}_{22}^{(i)}$ is generated earlier than $\mathbf{H}_{22}^{(j)}$ for $i < j$, equation (31) has to be computed from left to right. Obviously, we prefer to compute (31) from right to left such that only inner product computations are performed instead of 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.

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 (14) and the reflection vectors are sent to the right from boundary cells. From these observations, (31) can be computed in similar operations performed by the internal cells. A generalized architecture, as shown in Fig.2, 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.2. The $\mathbf{v}_n$ obtained at the output of RA is then backward propagated through the backward propagation array. From (14), each cell of this array performs the operation

$$\mathbf{H}_{22}^{(i)}\mathbf{v'}_n = \mathbf{v'}_n - \frac{2\mathbf{x}_{n,i}^{(i-1)}}{\sigma_{\mathbf{z}_i}^2}(\mathbf{x}_{n,i}^{(i-1)T} \cdot \mathbf{v'}_n), \tag{34}$$

where $\mathbf{v'}_n$ is an updated $\mathbf{v}_n$. This is a subset of the operations performed by the internal cell shown in (17). The residual vector is obtained from the top of the newly added 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. 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.

4 Conclusions

In this paper, we have shown that the Householder transformations can be implemented on a systolic array. 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 [6] can also be easily incorporated into the SBHT RLS array. A two-level pipelined implementation of the SBHT as well as its application to MVDR beamforming will be discussed in a future paper. From the results described in this paper, the Householder transformations becomes more promising for the pratical applications in real-time high throughput 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] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays", Proc. SPIE, Vol 298, Real Time Signal Processing IV, pp.298, 1981.

[3] G. H. Golub and C. F. Van Loan, **Matrix Computations,** Baltimore, MD: Johns Hopkins University Press, 1983.

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

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

[6] K.J.R. Liu and K. Yao,"Real-time algorithm-based fault-tolerance for QR recursive least-squares systolic array: a graceful degradation approach", submitted to IEEE Trans. on Computer, June, 1989.

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

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

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