A HIGHLY PARALLEL MULTICHANNEL FAST QRD-LS ADAPTIVE ALGORITHM

Athanasios A. Rontogiannis and Sergios Theodoridis
Department of Informatics
Division of Communications and Signal Processing
University of Athens
GR-157 71 Zografou, GREECE
e-mail:{tronto,stheodor}@di.uoa.gr

ABSTRACT

A new fast multichannel QR decomposition (QRD) least squares (LS) adaptive algorithm is presented in this paper. The algorithm deals with the general case of channels with different number of delay elements and is based exclusively on numerically robust orthogonal Givens rotations. The new scheme processes each channel separately and as a result it comprises scalar operations only. Moreover, the proposed algorithm is implementable on a very regular systolic architecture and offers substantially reduced computational complexity compared to previously derived multichannel fast QRD schemes.

1 INTRODUCTION

Multichannel least squares adaptive algorithms [1] are becoming increasingly popular due to their fast converging properties, and they find wide applications in diverse areas such as channel equalization, stereophonic echo cancellation, multidimentional signal processing, Volterra type nonlinear system identification, to name but a few. Among the various issues, characterizing the performance of an algorithm, those of computational complexity, parallelism and numerical robustness are of particular importance, in most applications. Especially the need for numerically robust schemes has led to the development of a class of algorithms based on the QR decomposition of the input data matrix, via the Givens rotations approach.
Multichannel fast QRD algorithms which spring from the single channel fast QRD schemes [2]-[4] have already been developed [6]-[8]. Both the cases of equal [6]-[7] and unequal [8] channel orders have been treated. Especially in [8] a novel channel decomposition technique is introduced, which makes possible the manipulation of channels of different orders. This channel decomposition procedure leads to a multichannel fast QRD algorithm consisting of l single channel fast QRD algorithms of length k, where l is the number of channels and k is the sum of the channel orders. These l single channel algorithms are interdependent and are executed sequentially, one after the other. The resulting algorithm is of $O(kl)$ computational complexity.
In this paper, a novel approach for deriving multichannel fast QRD algorithms is introduced. The new technique is based on the efficient time update of a particular vector quantity, which provides all the necessary for the LS error update, rotation parameters. A direct consequence of the new technique, is that explicit backward steps are essentially alleviated, a fact which simplifies the derivation procedure. The proposed methodology is also direct and insightful in that all algorithmic quantities involved have an obvious LS meaning and interpretation.
Based on the new technique a fast multichannel QRD algorithm is presented in this paper. The algorithm deals with the general case of unequal channel lengths. The channel partitioning used in [8] in the context of Volterra filtering, is also adopted here and the new algorithm consists of l single channel algorithms of the type of [5][1]. The proposed scheme can be implemented on a circular systolic architecture in which the single channel algorithms are executed in a pipelined fashion. In contrast, the channel decomposition based algorithms of [6],[7],[8] are strickly sequential for each time iteration. Moreover, compared to [8], the new algorithm offers reduced computational complexity in terms of both multiplications/divisions and square roots. This fact becomes apparent from the methodology adopted and is demonstrated with a specific example which concerns Volterra type nonlinear filtering.

2 FORMULATION OF THE PROBLEM

The standard exponentially weighted LS problem is that of selecting a $k \times 1$ coefficients' vector $\mathbf{c}(N)$ to satisfy the following optimization scheme

$$a_y(N) = \min_{\mathbf{c}(N)} \sum_{n=1}^{N} \lambda^{N-n} [y(n) - \mathbf{c}^T(N)\mathbf{u}(n)]^2 \tag{1}$$

λ stands for the usual forgetting factor with $0 \ll \lambda \leq 1$, $\mathbf{u}(n)$ is the $k \times 1$ input data vector and $y(n)$ is the desired response at time n, $n = 1, 2, \ldots, N$. The input-output information can be used to form the following $N \times (k+1)$ data matrix

$$[\mathbf{y}(N)|U(N)] = \Lambda(N) \begin{bmatrix} y(1) & \mathbf{u}^T(1) \\ y(2) & \mathbf{u}^T(2) \\ \vdots & \vdots \\ y(N) & \mathbf{u}^T(N) \end{bmatrix} \tag{2}$$

where $\Lambda(N) = diag[\lambda^{\frac{N-1}{2}}, \lambda^{\frac{N-2}{2}}, \ldots, 1]$. If $Q(N)$ stands for the orthogonal matrix which converts $U(N)$ into the $k \times k$ upper triangular form $\tilde{R}(N)$ then

$$Q(N)[\mathbf{y}(N)|U(N)] = \begin{bmatrix} \mathbf{p}(N) & \tilde{R}(N) \\ \mathbf{v}(N) & \bigcirc \end{bmatrix} \tag{3}$$

where $\mathbf{p}(N) \in \mathcal{R}^{k \times 1}$ and $\mathbf{v}(N) \in \mathcal{R}^{(N-k)\times 1}$. Since multiplication with an orthogonal matrix is norm preserving, it is

[1]Actually, the lattice type algorithm of [5] is slightly different

straightforward from (1),(2) and (3) that $\mathbf{c}(N)$ is given by

$$\tilde{R}(N)\mathbf{c}(N) = \mathbf{p}(N) \tag{4}$$

In a time varying environment, time update of $\tilde{R}(N)$ and $\mathbf{p}(N)$ is required, as new information becomes available. It turns out that all the necessary quantities for the update of these matrices can be obtained from the manipulation of the following vector term

$$\mathbf{g}(N+1) = \frac{\tilde{R}^{-T}(N)\mathbf{u}(N+1)}{\sqrt{\lambda}} \tag{5}$$

Indeed, if $\hat{Q}(N+1)$ is a sequence of k elementary Givens rotations which annul the elements of $-\mathbf{g}(N+1)$ as follows

$$\hat{Q}(N+1)\begin{bmatrix} -\mathbf{g}(N+1) \\ \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \\ \delta(N+1) \end{bmatrix} \tag{6}$$

then it can been shown [9] that this $\hat{Q}(N+1)$ updates $\tilde{R}(N)$ as well as $\mathbf{p}(N)$ according to the expression

$$\hat{Q}(N+1)\begin{bmatrix} \lambda^{1/2}\mathbf{p}(N) \\ \\ y(N+1) \end{bmatrix} = \begin{bmatrix} \mathbf{p}(N+1) \\ \\ \tilde{e}(N+1) \end{bmatrix} \tag{7}$$

The quantity $\delta(N+1)$ in (6) equals to the inverse of the angle normalized variable which relates the angle normalized error $\tilde{e}(N+1)$ to the a priori error $e(N+1)$ [5]

$$e(N+1) = \delta(N+1)\tilde{e}(N+1) \tag{8}$$

The efficient update of $e(N+1)$ is at the heart of our problem. It is defined as

$$e(N+1) = y(N+1) - \mathbf{c}^T(N)\mathbf{u}(N+1) \tag{9}$$

Finally note that the time update of the minimum squared error (energy) $a_y(N)$ is realized as follows [1]

$$a_y(N+1) = \lambda a_y(N) + [\tilde{e}(N+1)]^2 \tag{10}$$

In the next section, a new multichannel fast QRD algorithm is developed. It is shown that each step of the algorithm can be treated as an LS problem of the type described in this section, a fact which unifies the derivation procedure.

3 THE NEW ALGORITHM

Let us consider l input channels of lengths $k_1, k_2, \ldots, k_l$ respectively and $k = \sum_{r=1}^{l} k_r$. Without loss of generality we assume that $k_1 \geq k_2 \geq \cdots \geq k_l$. A critical point of our methodology is the selection of an appropriate partitioning of the input samples which appear in the input data vector $\mathbf{u}_k(N)$. Specifically, we choose the $k_1 - k_2$ most recent samples of the first channel to be the leading elements of $\mathbf{u}_k(N)$, followed by $k_2 - k_3$ pairs of samples of the first and second channel, followed by $k_3 - k_4$ triples of samples of the first three channels ... followed by k_l l-ples of samples of all channels. It is now straightforward that the position of the first (most recent) sample of the i-th channel is given by

$$m_i = \sum_{r=1}^{i-1} r(k_r - k_{r+1}) + i \qquad i = 1, 2, \ldots, l$$

Starting from $\mathbf{u}_k(N)$ we define the input data vectors: $\mathbf{u}_{k+1}^T(N+1) = [u_1(N+1), \mathbf{u}_k^T(N)]$ and $\mathbf{u}_{k+i}^T(N+1) = [u_i(N+1), \mathbf{u}_{k+i-1}^T(N+1)]S_i$ for $i = 2, 3, \ldots, l$. S_i is a permutation matrix which moves $u_i(N+1)$ to the m_i-th position after left shifting the first $m_i - 1$ elements of $\mathbf{u}_{k+i-1}^T(N+1)$. It can be easily verified that $\mathbf{u}_{k+l}^T(N+1) = [\mathbf{u}_k^T(N+1), u_1(N-k_1+1), \ldots, u_l(N-k_l+1)]$ that is, the first k elements of $\mathbf{u}_{k+l}^T(N+1)$ provide the input vector of the next time instant. The following input data matrices can now be defined for $i = 0, 1, \ldots, l$

$$U_{k+i}(N) = \Lambda(N)\begin{bmatrix} \mathbf{u}_{k+i}^T(1) \\ \mathbf{u}_{k+i}^T(2) \\ \vdots \\ \mathbf{u}_{k+i}^T(N) \end{bmatrix} \tag{11}$$

If $\tilde{R}_{k+i}(N)$ stands for the Cholesky factor of $U_{k+i}(N)$, the corresponding vectors $\mathbf{g}_{k+i}(N+1)$ can be expressed as

$$\mathbf{g}_{k+i}(N+1) = \frac{\tilde{R}_{k+i}^{-T}(N)\mathbf{u}_{k+i}(N+1)}{\sqrt{\lambda}} \tag{12}$$

The $\mathbf{g}_k(N)$ vector will also be the critical quantity here. From the discussion above and (12) it is not difficult to show that

$$\mathbf{g}_{k+l}(N+1) = \begin{bmatrix} \mathbf{g}_k(N+1) \\ \mathbf{g}^{(k)}(N+1) \end{bmatrix} \tag{13}$$

where $\mathbf{g}^{(k)}(N+1)$ consists of the last l elements of $\mathbf{g}_{k+l}(N+1)$. Time update of $\mathbf{g}_k(N)$ can now be realized according to the following scheme

$$\mathbf{g}_k(N) \rightarrow \mathbf{g}_{k+1}(N+1) \rightarrow \cdots \rightarrow \mathbf{g}_{k+l}(N+1)$$

This procedure involves l "forward" steps which will be described below. It is clear that because of (13) explicit backward steps are essentially avoided and thus our methodology only requires forward steps.

3.1 Forward step 1

From (11), the input data matrix $U_{k+1}(N)$ can be written as

$$U_{k+1}(N) = \begin{bmatrix} \lambda^{\frac{N-1}{2}}u_1(1) & \mathbf{0}^T \\ \lambda^{\frac{N-2}{2}}u_1(2) & \\ \vdots & U_k(N-1) \\ u_1(N) & \end{bmatrix} \tag{14}$$

The last expression defines a (forward) LS problem (see Eq.(2)) whose scalar desired response is the input of the first channel. From (14), the definition (12) and the input vector partition $\mathbf{u}_{k+1}^T(N+1) = [u_1(N+1), \mathbf{u}_k^T(N)]$ we easily deduce [5]

$$\mathbf{g}_{k+1}(N+1) = \hat{Q}_k^{f(1)}(N)\begin{bmatrix} r_k^{(1)}(N+1) \\ \\ \mathbf{g}_k(N) \end{bmatrix} \tag{15}$$

$r_k^{(1)}(N+1)$ stands for the k-th order normalized a priori error expressed as

$$r_k^{(1)}(N+1) = \frac{e_k^{(1)}(N+1)}{\sqrt{\lambda}\tilde{a}_k^{(1)}(N)} \tag{16}$$

$\hat{Q}_k^{f(1)}(N)$ is a sequence of k elementary Givens rotations which are produced by annihilating in a bottom-up procedure the elements of the rotated reference vector $\mathbf{p}_k^{(1)}(N)$

with respect to the square root, $\tilde{a}_k^{(1)}(N)$, of the energy (corresponding to the LS problem (14)) [5]

$$\hat{Q}_k^{f(1)}(N) \begin{bmatrix} \tilde{a}_k^{(1)}(N) \\ \\ \mathbf{p}_k^{(1)}(N) \end{bmatrix} = \begin{bmatrix} \tilde{a}_0^{(1)}(N) \\ \\ \mathbf{0} \end{bmatrix} \tag{17}$$

Note that the lower order quantities $r_j^{(1)}(N+1), \tilde{a}_j^{(1)}(N), j = k, k-1, \ldots, 1$ are successively computed from (15) and (17). The time update of $\mathbf{p}_k^{(1)}(N)$ is realized according to the formula

$$\hat{Q}_k^{(0)}(N) \begin{bmatrix} \lambda^{1/2}\mathbf{p}_k^{(1)}(N) \\ \\ u_1(N+1) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_k^{(1)}(N+1) \\ \\ \tilde{e}_k^{(1)}(N+1) \end{bmatrix} \tag{18}$$

$\hat{Q}_k^{(0)}(N)$ is a sequence of k Givens rotations which annul $-\mathbf{g}_k(N)$ with respect to 1 (Eqs. (6), (7)). During this procedure the quantities $\delta_j^{(0)}(N)$ $j = 1, 2, \ldots, k$ are successively generated. Furthermore, in (18) the angle-normalized errors $\tilde{e}_j^{(1)}(N+1), j = 1, 2, \ldots, k$ are calculated through the apllication of $\hat{Q}_k^{(0)}(N)$. Finally the annuling of $\mathbf{g}_{k+1}(N+1)$ obtained from (15)

$$\hat{Q}_{k+1}^{(1)}(N) \begin{bmatrix} -\mathbf{g}_{k+1}(N+1) \\ \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \\ \delta_{k+1}^{(1)}(N) \end{bmatrix} \tag{19}$$

yields the rotation angles of $\hat{Q}_{k+1}^{(1)}(N)$ as well as $\delta_j^{(1)}(N)$ $j = 1, 2, \ldots, k+1$, which are used in the second forward step.

3.2 Forward step i for $i = 2, \ldots, l$

The input data matrices $U_{k+i}(N)$ and $U_{k+i-1}(N)$ are related as follows

$$U_{k+i}(N) = \begin{bmatrix} \lambda^{\frac{N-1}{2}} u_i(1) & \\ \lambda^{\frac{N-2}{2}} u_i(2) & \\ \vdots & U_{k+i-1}(N) \\ u_i(N) & \end{bmatrix} S_i \tag{20}$$

Starting from (20), it can be shown that the following equation holds [10] (due to lack of space details are omitted here)

$$\mathbf{g}_{k+i}(N+1) = S_i^T \hat{Q}_{k+i-1}^{f(i)}(N) \begin{bmatrix} r_{k+i-1}^{(i)}(N+1) \\ \\ \mathbf{g}_{k+i-1}(N+1) \end{bmatrix}$$

$\hat{Q}_{k+i-1}^{f(i)}(N)$ is a sequence of $(k+i) - m_i$ Givens rotations that nullify the last $(k+i) - m_i$ elements of $\mathbf{p}_{k+i-1}^{(i)}(N)$ with respect to $\tilde{a}_{k+i-1}^{(i)}(N)$ in a bottom-up procedure. As a result, it is obvious from the last equation that the first $m_i - 1$ elements of $\mathbf{g}_{k+i}(N+1)$ and $\mathbf{g}_{k+i-1}(N+1)$ are identical. The time update of $\mathbf{p}_{k+i-1}^{(i)}(N)$ is realized through the application of a Givens matrix $\hat{Q}_{k+i-1}^{(i-1)}(N)$ in a way similar to (18). Moreover, the rotations required in the next forward step are computed from the equation

$$\hat{Q}_{k+i}^{(i)}(N+1) \begin{bmatrix} -\mathbf{g}_{k+i}(N+1) \\ \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \\ \delta_{k+i}^{(i)}(N) \end{bmatrix}$$

It has been stated, however, that the first $m_i - 1$ elements of $\mathbf{g}_{k+i}(N+1)$ and $\mathbf{g}_{k+i-1}(N+1)$ are equal. As a consequence, the first $m_i - 1$ rotation parameters of the i-th forward step have already been calculated in the previous forward steps. Such an observation can lead to a significant reduction of the computational complexity of the proposed algorithm depending on the channel orders. Especially the rotations produced at the l-th forward step pass to the filtering part of the algorithm as well as to the first forward step (of the next time instant) leading to a circular functionality. The new algorithm is shown in figure 1. Since k rotation parameters are essentially used in the filtering section, each forward procedure can be restricted to k elementary steps (instead of $k+i$). Furthermore, each procedure can be executed in a forward manner [10]. In figure 1, $\theta_j^{(i-1)}(N)$, $j = 1, 2, \ldots, k$ stand for the first k rotation angles of $\hat{Q}_{k+i-1}^{(i-1)}(N)$ while $\phi_j^{(i)}(N+1)$, $j = m_i, m_i+1, \ldots, k$ are the last $k - m_i + 1$ rotation angles of $\hat{Q}_{k+i-1}^{f(i)}(N+1)$. $p_j^{(i)}(N)$ is the j-th element of $\mathbf{p}_{k+i-1}^{(i)}(N)$ and $g_{j-1}^{(i)}(N+1)$ denotes the j-th element of $\mathbf{g}_{k+i}(N+1)$. In order to maintain a unified notation the following conventions are adopted for the l-th forward step : $\theta_j^{(l)}(N) = \theta_j^{(0)}(N+1)$, $\delta_j^{(l)}(N) = \delta_j^{(0)}(N+1)$ and $g_{j-1}^{(l)}(N)$ $= g_{j-1}^{(0)}(N+1)$, $j = 1, 2, \ldots, k$.

The multichannel algorithm of figure 1 is based exclusively on orthogonal Givens rotations. As a result, its numerical performance is expected to be favorable. For channels of equal orders ($k = lp$) the new scheme is of $O(pl^2)$ computational complexity which is is similar to that of other known channel decomposition based fast QRD schemes [6],[7]. In the general case of different channel orders, however, the proposed algorithm is of lower computational complexity if compared with the fast QRD scheme of [8] (which also treats unequal channel lengths). Specifically, the computational requirements of the two algorithms for the second order Volterra problem treated in [8], are shown in table 1. We observe that the new algorithm offers significant computational savings in terms of both the number of multiplications/divisions (11% less) and the number of square roots (20% less). This improvement can be even greater for a different application (in the higher order Volterra case, for instance).

The new algorithm can also be implemented on a very regular systolic architecture, as shown in figure 2. The architecture comprises l identical sections and each section consists of k blocks. The i-th section is excited from the i-th channel and essentially implements the i-th forward step. The blocks of the i-th section transfer the necessary values of $g^{(i)}, \theta^{(i)}$ and $\delta^{(i)}$ to the corresponding blocks of the $(i+1)$-th section. The last, l-th, section passes these quantities to the first channel leading to a circular implementation. It also sends the angles $\theta_j^{(l)}$ to the filtering section of the architecture (not shown in figure 2).

The proposed architecture is pipelinable at the order level, that is the throughput provided is constant, independent of k. This is achieved if we let the l inputs to be applied in a skewed manner from top-to-bottom. The skewing of the inputs ensures the synhronization of the different building blocks of the circular architecture, which provides the output error every l "clock cycles". Note that the fast channel decomposition based QRD algorithms of [6],[7],[8] are strickly sequential for each time iteration.

Alg.	Mult.'s/Div.'s	Sqrt.'s
[8]	$7.66L^3 + 30.5L^2 + O(L)$	$\frac{5}{6}L^3 + 3L^2 + O(L)$
New	$6.83L^3 + 23L^2 + O(L)$	$\frac{2}{3}L^3 + 2L^2 + O(L)$

Table 1: Comparison of complexities of multichannel fast QRD algorithms

4 CONCLUDING REMARKS

Least squares adaptive algorithms based on the QR decomposition of the input data matrix are very promising due to their numerically robust performance. In this paper, following a novel technique, a new multichannel fast QRD algorithm was developed. Besides its good numerical properties, the proposed algorithm exhibits some other nice features, such as fast convergence, low complexity and enhanced parallelism and pipelinability. As a consequence, the new algorithm is amenable to efficient implementations on systolic array architectures with short wordlengths and fixed-point arithmetic. As the number of applications which accept a multichannel formulation increases, schemes of the type presented in this paper appear to be appropriate algorithmic tools.

References

[1] N. Kalouptsidis and S. Theodoridis Eds., *Adaptive system identification and signal processing algorithms*, Engelwood Cliffs, NJ : Prentice Hall, 1993.

[2] J.M. Cioffi, "The fast adaptive ROTOR's RLS algorithm", *IEEE Trans. ASSP-38*, no. 4, pp. 631-653, Apr. 1990.

[3] I.K. Proudler, J.G. McWhirter, T.J. Shepherd, "Fast QRD-based algorithms for least squares linear prediction", *Proc. IMA Conference on Mathematics and Signal Processing*, Warwick, England, 12-15th Dec. 1988.

[4] P.A. Regalia, M.G. Bellanger, "On the duality between fast QR methods and lattice methods in least squares adaptive filtering", *IEEE Trans. SP-39*, no. 4, pp. 879-891, Apr. 1991.

[5] A.A. Rontogiannis, S. Theodoridis, "New fast inverse QR least squares adaptive algorithms", *Proc. IEEE ICASSP*, pp. 1412-1415, Detroit, MI, May 1995.

[6] M.G. Bellanger, P.A. Regalia, "The FLS-QR algorithm for adaptive filtering : the case of multichannel signals", *Signal Processing* 22, pp. 115-126, 1991.

[7] M.A. Syed, "QRD-based fast RLS multichannel adaptive algorithms", *Proc. IEEE ICASSP*, pp. 1837-1840, Toronto 1991.

[8] M.A. Syed, V.J. Mathews, "QR-decomposition based algorithms for adaptive Volterra filtering", *IEEE Trans. CAS-I-40*, no.6, pp. 372-382, June 1993.

[9] C.T. Pan, R.J. Plemmons, "Least squares modifications with inverse factorizations : parallel implications", *J. Compt. Appl. Math.* vol. 27, pp. 109-127, 1989.

[10] A.A. Rontogiannis, S. Theodoridis, "Multichannel fast QRD-LS adaptive filtering: New technique and algorithms", submitted to *IEEE Trans. on SP.*

```
$\delta_0^{(1)}(N) = 1$; $\tilde{e}_0(N+1) = y(N+1)$;
$g_0^{(1)}(N+1) = \frac{u_1(N+1)}{\sqrt{\lambda}\tilde{a}_0^{(1)}(N)}$;
$\tilde{a}_0^{(1)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_0^{(1)}(N))^2 + (u_1(N+1))^2}$;
for $i = 1 : l$,
 $\tilde{e}_0^{(i)}(N+1) = u_i(N+1)$;
 for $j = 1 : k$,
   $p_j^{(i)}(N+1) = \lambda^{1/2}\cos[\theta_j^{(i-1)}(N)]p_j^{(i)}(N) + \sin[\theta_j^{(i-1)}(N)]\tilde{e}_{j-1}^{(i)}(N+1)$;
   $\tilde{e}_j^{(i)}(N+1) = \cos[\theta_j^{(i-1)}(N)]\tilde{e}_{j-1}^{(i)}(N+1) - \lambda^{1/2}\sin[\theta_j^{(i-1)}(N)]p_j^{(i)}(N)$;
   If $j \geq m_i - 1$,
    $\tilde{a}_j^{(i)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_j^{(i)}(N))^2 + (\tilde{e}_j^{(i)}(N+1))^2}$;
    $r_j^{(i)}(N+1) = \frac{\tilde{e}_j^{(i)}(N+1)\delta_j^{(i-1)}(N)}{\sqrt{\lambda}\tilde{a}_j^{(i)}(N)}$;
    If $j = m_i - 1$,
     $g_j^{(i)}(N+1) = r_j^{(i)}(N+1)$;
   If $j > m_i - 1$,
    $g_j^{(i)}(N+1) = -\sin[\phi_j^{(i)}(N)]r_j^{(i)}(N+1) + \cos[\phi_j^{(i)}(N)]g_{j-1}^{(i-1)}(N+1)$;
    $\cos[\phi_j^{(i)}(N+1)] = \frac{\tilde{a}_j^{(i)}(N+1)}{\tilde{a}_{j-1}^{(i)}(N+1)}$;
    $\sin[\phi_j^{(i)}(N+1)] = \frac{p_j^{(i)}(N+1)}{\tilde{a}_{j-1}^{(i)}(N+1)}$;
    $\delta_j^{(i)}(N) = \sqrt{(\delta_{j-1}^{(i)}(N))^2 + (g_{j-1}^{(i)}(N+1))^2}$;
    $\cos[\theta_j^{(i)}(N)] = \frac{\delta_{j-1}^{(i)}(N)}{\delta_j^{(i)}(N)}$;
    $\sin[\theta_j^{(i)}(N)] = \frac{g_{j-1}^{(i)}(N+1)}{\delta_j^{(i)}(N)}$;
 end; { $j$-loop }
end; { $i$-loop }
for $j = 1 : k$,
 $p_j(N+1) = \lambda^{1/2}\cos[\theta_j^{(0)}(N+1)]p_j(N) + \sin[\theta_j^{(0)}(N+1)]\tilde{e}_{j-1}(N+1)$;
 $\tilde{e}_j(N+1) = \cos[\theta_j^{(0)}(N+1)]\tilde{e}_{j-1}(N+1) - \lambda^{1/2}\sin[\theta_j^{(0)}(N+1)]p_j(N)$;
end;
$e_k(N+1) = \delta_k^{(0)}(N+1)\tilde{e}_k(N+1)$;
```

Figure 1: The new multichannel fast QRD algorithm

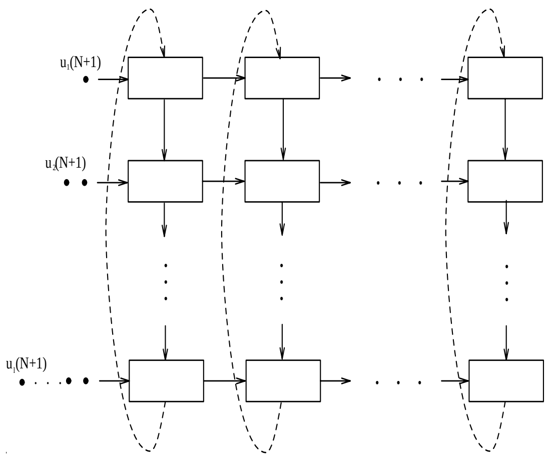

Figure 2: A systolic architecture for the implementation of the new algorithm