PIPELINED CORDIC BASED QRD-MVDR ADAPTIVE BEAMFORMING *

Jun Ma[1] *Keshab K. Parhi*[1] *Ed F. Deprettere*[2]

[1] Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, U.S.A.
[2] Department of Electrical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands

ABSTRACT

Cordic based QRD-MVDR adaptive beamforming algorithms possess desirable properties for VLSI implementation such as regularity and good finite-word length behavior. But this algorithm suffers from speed limitation constraint due to the presence of recursive operations in the algorithm. In this paper, a fine-grain pipelined Cordic based QRD-MVDR adaptive beamforming algorithm is developed using the matrix lookahead technique. The proposed architecture can operate at arbitrarily high sample rates, and consists of only Givens rotations which can be mapped onto a Jacobi specific dataflow processor. It requires a complexity of $O(M(p^2 + Kp))$ Givens rotations per sample time, where p is the number of antenna elements, K is the number of look direction constrains, and M is the pipelining level.

1. INTRODUCTION

QR decomposition based minimum variance distortionless response (QRD-MVDR) adaptive beamforming algorithm [1] possess desirable properties for VLSI implementation such as regularity and good finite-word length behavior. However, the speed or sample rates of the algorithm's implementation is limited by time required by the individual cells. The computation of each cell can't be pipelined to finer level (such as bit or mult-bit level) due to the presence of recursive operations in the cell. In many adaptive beamforming applications, very high sample rates would be required, and the QRD-RLS algorithm may not be able to operate at such high sample rates.

To increase the speed of the QRD-MVDR, lookahead techniques [2] or block processing techniques [3] can be applied. The so-called STAR rotation developed in [4] and relaxed lookahead technique developed in [5] allow fine-grain pipelining with little hardware overhead. However, this is achieved at the cost of degradation of filtering performance due to the approximations in the algorithm. Both algorithms in [4] and [5] are based on multiply-add arithmetic. Recently, matrix lookahead technique was developed in [6] to achieve fine-grain pipelining in QRD-RLS adaptive filtering. It is an exact lookahead and is based on Cordic arithmetic. Furthermore, the transformation does not alter algorithm orthogonality, leading to good finite-word length behavior, which is attractive for VLSI implementation. In this paper, a fine-grain pipelined QRD-MVDR adaptive beamforming algorithm is developed using the matrix lookahead technique. The proposed architecture can operate at arbitrarily high sample rates, and consists of only Givens rotations which may be mapped onto a Cordic based Jacobi specific dataflow processor [7] [8]. The matrix lookahead transformation, in this paper, is developed in a systematic way from the block processing point of view which is slightly different from the derivation in [6], although they bear the same mathematical roots. Through this derivation, it is also shown how pipelining and block processing in adaptive digital filters are linked by the lookahead transformation.

*THIS RESEARCH WAS SUPPORTED IN PARTS BY NSF UNDER GRANT NUMBER MIP-9258670.

2. QRD-MVDR ALGORITHM

The minimum variance distortionless response (MVDR) adaptive beamforming problem may be summarized as follows. At each sample time instant n, evaluate *a posteriori* residual

$$e^{(k)}(n) = \mathbf{u}^T(n)\,\mathbf{w}^{(k)}(n) \tag{1}$$

where $\mathbf{u}(n)$ is the p-element vector of signal samples received by the array at time instant n, and $\mathbf{w}^{(k)}(n)$ is the p-element vector of weights which minimize the quantity

$$\xi^{(k)}(n) = \| \epsilon^{(k)}(n) \| = \| A(n)\,\mathbf{w}^{(k)}(n) \|, \tag{2}$$

subject to a linear equality constraint of the form

$$\mathbf{c}^{(k)\,T}\,\mathbf{w}^{(k)}(n) = \mu^{(k)}. \tag{3}$$

where $A(n) = \left[\begin{array}{cccc} \mathbf{u}(1) & \mathbf{u}(2) & \cdots & \mathbf{u}(n) \end{array}\right]^T$ is the input data matrix, and $\mathbf{c}^{(k)}$ is the kth steering vector which represents the kth desired look direction, and $\mu^{(k)}$ is the corresponding beamforming gain which is usually a constant.

The solution to this constrained least squares minimization problem is given by the following well known formula

$$\mathbf{w}^{(k)}(n) = \mu^{(k)}\,\frac{\Phi^{-1}(n)\,\mathbf{c}^{(k)}}{\mathbf{c}^{(k)\,T}\,\Phi^{-1}(n)\,\mathbf{c}^{(k)}} \tag{4}$$

where Φ is the covariance matrix defined by

$$\Phi(n) = A^T(n)\,A(n). \tag{5}$$

Here, we assume all the data are real. The extension to the complex case is straightforward. Also, for clarity purpose, the forgetting factor in the least squares estimation formulation is ignored during our discussion.

Assuming that a QR decomposition has been carried out on the data matrix $A(n)$ so that

$$Q(n)\,A(n) \;=\; \left[\begin{array}{c} R(n) \\ 0 \end{array}\right] \tag{6}$$

where $R(n)$ is a p-by-p upper triangular matrix, then it follows that

$$\Phi(n) \;=\; R^T(n)\,R(n) \tag{7}$$

and so $R(n)$ is the Cholesky square root factor of the covariance matrix $\Phi(n)$. Equation (4) may therefore be written in the form

$$\begin{aligned} \mathbf{w}^{(k)}(n) &= \mu^{(k)}\,\frac{R^{-1}(n)\,R^{-T}(n)\,\mathbf{c}^{(k)}}{\mathbf{c}^{(k)T}\,R^{-1}(n)\,R^{-T}(n)\,\mathbf{c}^{(k)}} \\ &= \mu^{(k)}\frac{R^{-1}(n)\,\mathbf{a}^{(k)}(n)}{\parallel \mathbf{a}^{(k)}(n) \parallel^2} \end{aligned} \tag{8}$$

where

$$\mathbf{a}^{(k)}(n) \;=\; R^{-T}(n)\,\mathbf{c}^{(k)}. \tag{9}$$

It follows that the *a posterior* residual at time instant n is given by

$$e^{(k)}(n) \;=\; \mathbf{u}^T(n)\,\mathbf{w}^{(k)}(n) \;=\; \mu^{(k)}\,\frac{\mathbf{u}^T(n)\,R^{-1}(n)\,\mathbf{a}^{(k)}(n)}{\parallel \mathbf{a}^{(k)}(n) \parallel^2}. \tag{10}$$

Let

$$e'^{(k)}(n) \;=\; \mathbf{u}^T(n)\,R^{-1}(n)\,\mathbf{a}^{(k)}(n). \tag{11}$$

That is $e'^{(k)}(n)$ is a scaled version of $e^{(k)}(n)$ with the scaling factor $\mu^{(k)}/\parallel \mathbf{a}^{(k)}(n) \parallel^2$.

The QR decomposition of the data matrix $A(n)$ can be implemented in a recursive manner. With each incoming data sample set, a new row $\mathbf{u}^T(n)$ is appended to the data matrix $A(n-1)$ to yield $A(n)$. A set of p Givens rotations are determined to null the last row of $A(n)$. Thus the triangular matrix $R(n-1)$ gets updated to $R(n)$. The QR update procedure can be described by the following equation

$$\left[\begin{array}{c} R(n) \\ \mathbf{0}_p^T \end{array}\right] = Q(n) \left[\begin{array}{c} R(n-1) \\ \mathbf{u}^T(n) \end{array}\right]. \tag{12}$$

After some algebraic manipulations, it can be shown that the orthogonal matrix Q in (12) also updates the auxiliary vector $\mathbf{a}^{(k)}$ in (9) [1]. Therefore, the QR update for the MVDR adaptive beamforming algorithm can be summarized as follows.

$$\left[\begin{array}{ccc} R(n) & \mathbf{a}^{(k)}(n) & \mathbf{g}(n) \\ \mathbf{0}_p^T & \alpha^{(k)}(n) & \gamma(n) \end{array}\right] = Q(n) \left[\begin{array}{ccc} R(n-1) & \mathbf{a}^{(k)}(n-1) & \mathbf{0}_p \\ \mathbf{u}^T(n) & 0 & 1 \end{array}\right] \tag{13}$$

The insertion of the third column in (13) is used to generate the converting factor $\gamma(n)$. The scaled residual $e'^{(k)}(n)$ can be obtained using the following equation [9]

$$e'^{(k)}(n) \;=\; -\,\alpha^{(k)}(n)\,\gamma(n). \tag{14}$$

An efficient flow graph representation of the Cordic based QRD-MVDR adaptive beamforming algorithm is shown in Fig. 1. In this figure, the circle and square cells denote Cordic operations with circle cells operating in vectoring mode and square cells operating in rotating mode. The circle cell with *right-angle* and letter G inside denote a Gaussian rotation. Its functionality is shown in the Figure. The algorithm complexity is $O(p^2 + Kp)$ Givens rotations per sample time, where p is the size of the antenna array and K is the number of look direction constraints.

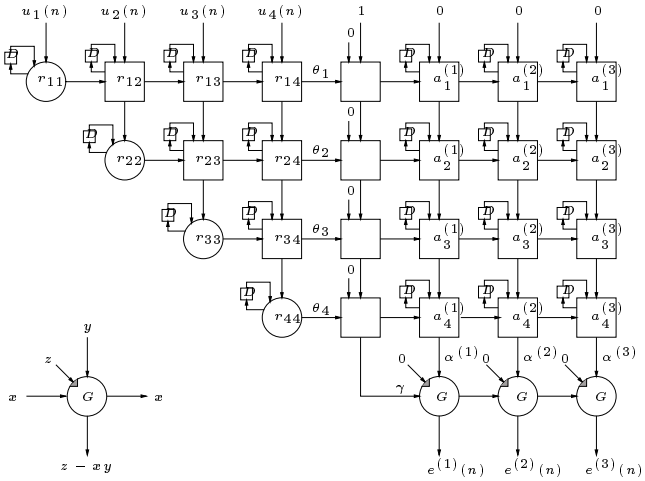

Figure 1. Flow graph representation of QRD-MVDR adaptive beamforming algorithm.

3. PIPELINING OF QRD-MVDR

From Fig. 1, we see that the QRD-MVDR algorithm can be easily pipelined at cell level after applying *cut-set* pipelining. The speed or sample rates of the algorithm is, however, limited by time required by the individual cells. The computation of each cell can't be pipelined at finer level (such as bit or multi-bit level) due to the presence of recursive operations in the cell as described algebraically by (13). In order to achieve arbitrarily high sample rates, the matrix lookahead transformation is considered.

3.1. Matrix Lookahead Overview

The matrix lookahead transformation was recently developed to achieve fine-grain pipelining in Givens rotation based recursive least squares adaptive filtering [6]. It is an exact lookahead and is based on Cordic arithmetic. One of the important property of this transformation is that it can transform an *orthogonal sequential* recursive algorithm to an equivalent *orthogonal concurrent* one by creating additional concurrency in the algorithm. The resulting transformed algorithm possesses both pipelinability and good finite word

length behavior which are attractive for VLSI implementations. The matrix lookahead transformation for Givens rotation based adaptive filtering algorithms are summarized in the following two-step procedure.

1. Formulate block updating form of the recursive operations with block size equal to the pipelining level M.
2. Choose a sequence of Givens rotations to perform the updating in such a way that it first operates on the block data and then updates the recursive variables. The aim is to reduce the computational complexity of a block update inside the feedback loop to the same complexity as a single-step update.

3.2. Block Processing Formulation of QRD-MVDR Algorithm

The recursive updating formula for the QRD-MVDR algorithm is given in (13). Its block updating form with block size M is given as follows

$$\begin{bmatrix} R(n) & \mathbf{a}^{(k)}(n) & \mathbf{g}(n) \\ O_{M\times p} & \boldsymbol{\alpha}^{(k)}(n) & \boldsymbol{\gamma}(n) \end{bmatrix} = Q(n) \begin{bmatrix} R(n-M) & \mathbf{a}^{(k)}(n-M) & \mathbf{0}_p \\ U_M^T(n) & \mathbf{0}_M & \boldsymbol{\delta}_M \end{bmatrix}, \quad (15)$$

where $U_M^T(n)$ is an M-by-p matrix defined as

$$U_M^T(n) = \begin{bmatrix} \mathbf{u}(n-M+1) & \cdots & \mathbf{u}(n-1) & \mathbf{u}(n) \end{bmatrix}^T,$$

$O_{M\times p}$ and $\mathbf{0}_M$ denote M-by-p null matrix and M-by-1 null vector, respectively. $\boldsymbol{\alpha}^{(k)}(n)$ and $\boldsymbol{\gamma}(n)$ are M-by-1 vectors, and $\boldsymbol{\delta}_M$ is a M-by-1 constant vector defined as

$$\boldsymbol{\delta}_M = [0, \cdots, 0, 1]^T.$$

The scaled *a posteriori* residual $e'^{(k)}(n)$ is given as

$$e'^{(k)}(n) = -\boldsymbol{\alpha}^{(k)^T}(n) \cdot \boldsymbol{\gamma}(n). \quad (16)$$

The derivations of equations (15) and (16) are omitted here due to lack of space. Notice that the $Q(n)$ matrix in (15) is different from the $Q(n)$ in (13), though we use the same notation here.

3.3. Determination of the Givens Rotation Sequence

Now, we determine a sequence of Givens rotations, whose product form the orthogonal transformation matrix $Q(n)$ in (15), to annihilate the block input data matrix $U_M^T(n)$. The order of the Givens rotations is chosen such that the input data is pre-processed and block-data update is finished in the same complexity as a single-data update. We illustrate this procedure using a relatively small size example. In (15), notice that only the upper triangular matrix $R(n-M)$ and the input data matrix $U_M^T(n)$ is used to determine the sequence of Givens rotations. The determined rotations are then used to update $\mathbf{a}^{(k)}$ and generate $\boldsymbol{\gamma}$. Therefore, in our illustration, only $R(n-M)$ and $U_M^T(n)$ are included for clarity purpose. Let $p = 4$ and $M = 3$. The block QR update formula for adaptive MVDR algorithm can be written as follows

$$\begin{bmatrix} r & r & r & r & \\ & r & r & r & (n) \\ & & r & r & \\ & & & r & \\ 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \end{bmatrix} = Q \begin{bmatrix} r & r & r & r & \\ & r & r & r & (n-3) \\ & & r & r & \\ & & & r & \\ u & u & u & u & (n-2) \\ u & u & u & u & (n-1) \\ u & u & u & u & (n) \end{bmatrix} \quad (17)$$

Let $G(i,j)$ denote a $(M+p)$-by-$(M+p)$ plane rotation, which is 7-by-7 in our example.

$$G(i,j)(n) = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{bmatrix} I & & & & & & \\ & c & & s & & & \\ & & I & & & & \\ & -s & & c & & & \\ & & & & I & & \end{bmatrix} \\ \quad i \qquad\quad j \end{array} \quad (18)$$

A sequence of Givens rotations can be chosen as follows.

$$\begin{array}{l} G(6,7) \longrightarrow G(5,6) \longrightarrow G(1,5) \longrightarrow \\ G(6,7) \longrightarrow G(5,6) \longrightarrow G(2,5) \longrightarrow \\ G(6,7) \longrightarrow G(5,6) \longrightarrow G(3,5) \longrightarrow \\ G(6,7) \longrightarrow G(5,6) \longrightarrow G(4,5) \end{array} \quad (19)$$

Each of the above four rows from top to down annihilates one column of the data matrix $U_M^T(n)$ from left to right, accordingly. In this case, the input data samples are pre-processed and the r elements are updated only at the last step. The signal flow graph of a typical r element update is shown in Fig. 2. Therefore, without increasing the loop computational complexity, we increase the number of delay elements in the feedback loop from one delay element to three delay elements. These three delay elements can then be redistributed around the loop using the *retiming* technique [10] to achieve pipelining by 3-level. The two Cordic processors outside the feedback loop are the computation overhead due to the lookahead transformation. Since they are feed-forward, cutset pipelining can be applied to speed them up. Furthermore, the overhead Cordic processors outside the loop can be arranged in a *tree* structure to explore the parallelism and reduce overall latency. The

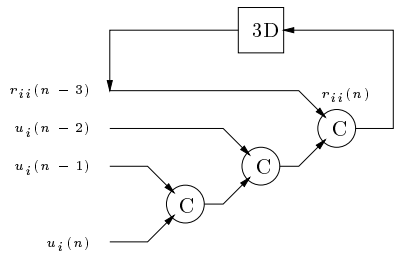

Figure 2. Matrix lookahead transformation.

Givens rotation sequence which leads to the matrix lookahead transformation is not unique, another candidate may

be

$$
\begin{aligned}
&G(6,7) \longrightarrow G(5,6) \longrightarrow G(6,7) \longrightarrow \\
&G(1,5) \longrightarrow G(5,6) \longrightarrow G(6,7) \longrightarrow \\
&G(2,5) \longrightarrow G(5,6) \longrightarrow G(6,7) \longrightarrow \\
&G(3,5) \longrightarrow G(5,6) \longrightarrow G(4,5)
\end{aligned} \tag{20}
$$

The above sequence corresponds to annihilating the data matrix $U_M^T(n)$ in a *QR decomposition* manner.

The lookahead transformation shown in Fig. 2 can be exploited either in the form of block processing [3] or pipelining [2].

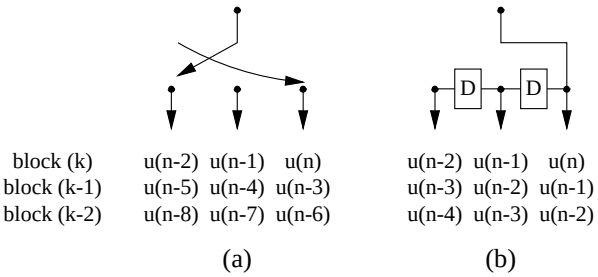

Figure 3. Serial to parallel conversion for (a) Block processing and (b) Pipelining.

In a block processing realization of size M, the input samples are processed in a block manner through a *serial-to-parallel* converter shown in Fig. 3(a). Because the input samples in two consecutive blocks are not overlapped, one filtering output is obtained every M sample periods. In order to obtain the rest outputs, either extra $M-1$ duplicated operations are needed, which is usually very expensive, or apply *incremental block processing* technique to reduce computational complexity [3].

In a pipelined realization, the input samples are also processed in the block manner but through a tapped delay line as shown in Fig. 3(b). In this case, consecutive block samples are shift–overlapped, thus all filtering output can be obtained consecutively. The final pipelined QRD-MVDR adaptive beamforming architecture with pipelining level 3 is shown in Fig. 4. In this figure, all cell notations follow the notations in Fig. 1 except that they are compound versions. The internal structure of each compound cell is shown at the bottom part of Fig. 4. Notice that, compared to Fig. 1, the 3-level pipelined architecture tripled the number of Cordic cells and communication bandwidth which is linear with respect to the pipelining level. Thus, the total complexity is $O(M(p^2+Kp))$ Givens rotations per sample time, where p is the number of antenna elements, K is the number of look direction constrains, and M is the pipelining level. The pipelined QRD-MVDR architecture shown in Fig. 4 can be mapped onto a Cordic based Jacobi specific dataflow processor [8].

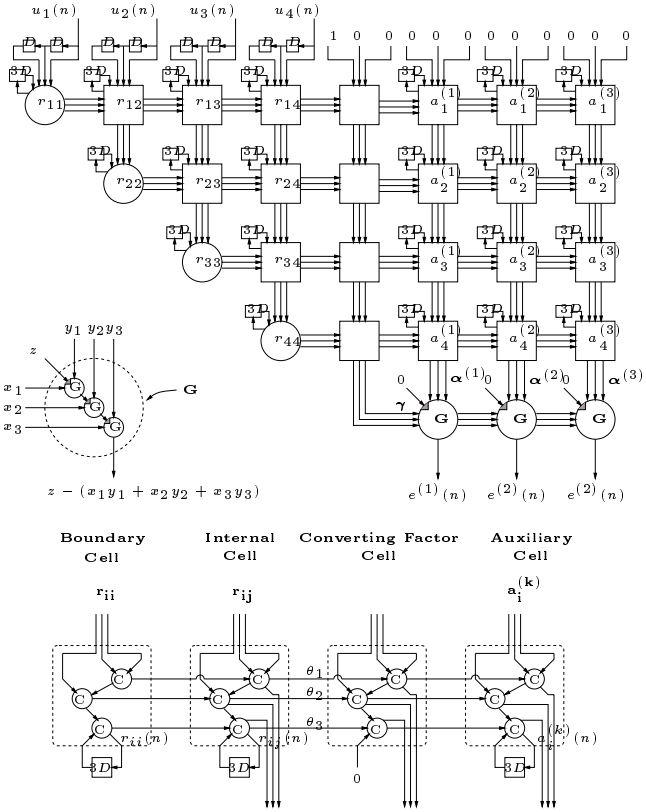

Figure 4. A 3–level pipelined Cordic based QRD-MVDR adaptive beamforming architecture.

REFERENCES

[1] J.G. McWhirter and T.J. Shepherd, "Systolic array processor for MVDR beamforming", *in IEE Proceedings*, vol. 136, pp. 75–80, April 1989.

[2] K.K. Parhi and D.G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters–Part I: Pipelining using scattered look-ahead and decomposition", *IEEE Trans. on ASSP*, vol. 37, pp. 1118–1134, July 1989.

[3] K.K. Parhi and D.G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters–Part II: Pipelined incremental block filtering", *in IEEE Trans. on ASSP*, vol. 37, pp. 1099–1117, July 1989.

[4] K.J. Raghunath and K.K. Parhi, "Pipelined RLS Adaptive Filtering Using Scaled Tangent Rotations (STAR)", *IEEE Transcations on SP*, vol. 40, pp. 2591–2604, October 1996.

[5] N.R. Shanbhag and K.K. Parhi, *Pipelined Adaptive Digital Filters*, Kluwer Academic Publishers, 1994.

[6] J. Ma, E.F. Deprettere, and K.K. Parhi, "Pipelined Cordic based QRD-RLS adaptive filtering using matrix lookahead", *in IEEE Workshop on Signal Processing Systems*, November 1997.

[7] G.J. Hekstra and E.F. Deprettere, "Floating Point Cordic", *in Proceedings ARITH 11, Windsor, Ontario*, 1993.

[8] E. Rijpkema, G. Hekstra, E. Deprettere, and J. Ma, "A strategy for determining a Jacobi Specific Dataflow Processor", *in IEEE International Conf. on ASAP*, July 1997.

[9] J.G. McWhirter, "Recursive least-squares minimization using a systolic array", *in Proc. SPIE, Real Time Signal Processing VI*, vol. 431, pp. 105–112, 1983.

[10] C.E. Leiserson, F. Rose, and J. Saxe, "Optimizing synchronous circuitry by retiming", *in Proc. Third Caltech Conf. VLSI, Pasadena, CA*, pp. 87–116, March 1983.