I. Abstract

This report focuses on a hierarchical approach for pipelining the large CORDIC [1](Coordinate rotation digital computer)-based systolic array of a QR decomposition-based recursive least square algorithm (QRD-RLS) adaptive filter to a small fixed size array. Cordic-based QRD-MVDR adaptive beamforming algorithm possesses desirable properties for VLSI implementation such as regularity and good finite-word length behavior. [2] But this algorithm suffers from speed limitation constraint due to the presence of recursive operations in the algorithm. With the annihilation-reordering look-ahead transformation in [3], the iteration bound of a QRD-RLS adaptive filter can be reduced linearly with respect to the look-ahead factor. It can be found in this paper that how a combination of look-ahead, pipelined, and folding transformations can lead to a large increase in throughput and large reduction in area or power consumption. Therefore, this design technique is of great significance for application-specific IC chip design, high-level hardware synthesis, and special-purpose processor design. The optimally designed QRD-RLS adaptive filter can be used for adaptive digital beamforming applications, which play an important role in radar, sonar, and mobile/wireless communication system. In this report, we repeat and verify the simulation of Cordic-based QRD-MVDR adaptive beamforming in Matlab.

II. Introduction of QRD-RLS Adaptive Filters

In this section, the QRD-RLS algorithms, and their applications to adaptive digital beamforming are summarized.

A. Standard QRD-RLS Algorithm:

This RLS algorithm recursively updates the estimation for a least squares minimization problem. The computation starts with unknown initial conditions and uses the new data samples to update the old estimation. Rather than dealing with stationary stochastic process, the RLS algorithm will face nonstationary process most of time. Mathematically, RLS algorithm can be described as following a system with a linear combiner of K-tap. The cost function for this LS minimization problem at time $n$ is defined by [4]
\[ E(n) = \sum_{i=1}^{n} \lambda^{n-i} |c(i)|^2 \]

Where \( \lambda \) is called the forgetting factor and has a value of \( 0 < \lambda < 1 \), \( e(i) \) is the system error at time instance \( i \) and is defined by
\[
  c(i) = d(i) - u^T(i)w(n). 
\]

\( d(n) \) is the desired data at time instance \( n \), \( w(n) \) is the tap-weight vector of linear combiner at time instance \( n \) and is defined by
\[
  w(n) = [w_0(n) \ w_1(n) \ \cdots \ w_{K-1}(n)]^T
\]
and \( u(i) \) is the tap-input vector to the linear combiner at time instance \( i \) and is defined by
\[
  u(i) = [u_0(i) \ u_1(i) \ \cdots \ u_{K-1}(i)]^T.
\]

The optimization is to choose the tap-weight vector \( w(n) \) to minimize the cost function. Let the \( n \)-by-\( K \) data matrix \( A(n) \) be defined as
\[
  A(n) = \begin{bmatrix}
  \mathbf{u}(1) & \mathbf{u}(2) & \cdots & \mathbf{u}(n) \\
  \mathbf{u}(1) & \mathbf{u}(2) & \cdots & \mathbf{u}(1) \\
  \vdots & \vdots & \ddots & \vdots \\
  \mathbf{u}(n) & \mathbf{u}(n) & \cdots & \mathbf{u}(n) 
\end{bmatrix}
\]

And \( n \)-by-1 desired data vector \( \mathbf{d}(n) \) be defined as
\[
  \mathbf{d}(n) = \begin{bmatrix}
  d(1) \\
  d(2) \\
  \vdots \\
  d(n)
\end{bmatrix}^T.
\]

Then the \( n \)-by-1 system error vector \( \mathbf{e}(n) \) equals
\[
  \mathbf{e}(n) = \mathbf{d}(n) - A(n)w(n).
\]

Let the \( n \)-by-\( n \) exponential weight matrix \( \mathbf{F}(n) \) be defined as
\[
  \mathbf{F}(n) = \text{diag}[\lambda^{n-1} \ \lambda^{n-2} \ \cdots \ 1],
\]
Then the cost function can be redefined in matrix format by
\[ E(n) = e^H(n)A(n)e(n) \]
\[ = ||\Lambda^{1/2}(n)e(n)||^2 \]
\[ = ||\Lambda^{1/2}(n)d(n) - \Lambda^{1/2}(n)A(n)w(n)||^2. \]

Under the assumption that \( n > K \) and \( A(n) \) has full column rank, represents the LS minimization of an overdetermined system and has a unique solution. There are two approaches to solve this minimization problem: normal equation method and QR-decomposition method. The QR decomposition method has superior numerical stability over the RLS derived from the matrix inversion lemma. The QR-decomposition method starts from the data matrix using unitary transformation. Recall that the two-norm is preserved under unitary transformations. For a given unitary matrix \( Q(n) \), the cost function can be equivalently expressed as

\[ E(n) = ||Q(n)\Lambda^{1/2}(n)e(n)||^2 \]
\[ = ||Q(n)\Lambda^{1/2}(n)d(n) - Q(n)\Lambda^{1/2}(n)A(n)w(n)||^2. \]

To solve the minimization problem defined by cost function, the unitary \( Q(n) \) is chosen to triangulate the exponentially weighted data matrix.

\[ Q(n)\Lambda^{1/2}(n)A(n) = \begin{bmatrix} R(n) \\ 0 \end{bmatrix} \]

Where \( R(n) \) is \( K \)-by-\( K \) complex upper triangular matrix and 0 is an \((n-K)\)-by-\( K \) null matrix. The desired signal vector, after being transformed, is defined by

\[ Q(n)\Lambda^{1/2}(n)d(n) = \begin{bmatrix} p(n) \\ v(n) \end{bmatrix} \]

Where \( p(n) \) is a \( K \)-by-1 vector and \( v(n) \) is an \((n-K)\)-by-1 vector. Then we can rewrite the cost function

\[ E(n) = \left\| \begin{bmatrix} p(n) \\ v(n) \end{bmatrix} - \begin{bmatrix} R(n) \\ 0 \end{bmatrix}w(n) \right\|^2 \]
\[ = \left\| \begin{bmatrix} p(n) - R(n)w(n) \\ v(n) \end{bmatrix} \right\|^2. \]

It is obvious that the least squares estimation \( w(n) \) for the tap-weight
vector must satisfy
\[ p(n) - R(n)\hat{w}(n) = 0_{K \times 1} \]

\[ \hat{w}(n) = R^{-1}(n)p(n) \]

The unitary matrix \( Q(n) \), the upper triangular matrix \( R(n) \), and the vector \( p(n) \) can be recursively computed using

\[
\begin{bmatrix}
R(n) & p(n) \\
0_{(n-K-1) \times K} & 0_{(n-K-1) \times 1} \\
0_{1 \times K} & \alpha(n)
\end{bmatrix}
\begin{bmatrix}
\lambda^{1/2}R(n-1) \\
0_{(n-K-1) \times K} \\
u^T(n) \\
d(n)
\end{bmatrix}
= \hat{Q}(n)\begin{bmatrix}
\alpha(n) \\
\gamma^{1/2}(n) \\
u^T(n) \\
d(n)
\end{bmatrix}
\]

\[ Q(n) = \hat{Q}(n)\begin{bmatrix}
Q(n-1) & 0 \\
0 & 1
\end{bmatrix} \]

Therefore, optimal \( w(n) \) can be obtained. But in some application such as beamforming and linear prediction, \( e(n) \) is the desired signal. By expanding, we can obtain \( e(n) \) directly without explicitly extracting the tap-weight vector. This result is summarized in following equation.

\[
\begin{bmatrix}
R(n) & p(n) & R^{-1}(n)u(n) \\
0_{1 \times K} & \alpha(n) & (\gamma^{1/2}(n))^* \\
u^T(n) & & d(n)
\end{bmatrix}
= \hat{Q}(n)\begin{bmatrix}
\lambda^{1/2}R(n-1) \\
\lambda^{1/2}p(n-1) \\
u^T(n) \\
d(n)
\end{bmatrix}
\]

Where \( Q(n) \) unitary matrix that triangular the right side matrix;
\( \hat{\alpha}(n) \) angle normalized estimation error;
\( \tilde{\alpha}(n) \) conversion factor

The QR decomposition can be computed using any kind of unitary transformations, such as the Householder, block Householder, Given rotation, fast Givens rotation, classical Gram-Schmidt, and modified
Gram-Schmidt. The Givens rotation is attractive due to its superior numerical properties, high concurrency, and regularity for implementation. The Givens rotation can be expressed in following form

\[
\mathbf{G}(\theta_i(n)) = \begin{bmatrix}
\mathbf{I}(\theta_i) & : & : \\
: & \cos(\theta_i(n)) & \sin(\theta_i(n)) \\
\vdots & \vdots & \vdots \\
\cos(\theta_i(n)) & \sin(\theta_i(n)) & \mathbf{I}(K-i-1)
\end{bmatrix}
\]

\[(i = 0, 1, 2, \ldots, K - 1) \tag{22}\]

Thus, the matrix \(\mathbf{Q}(n)\) can be constructed as

\[
\mathbf{Q}(n) = \mathbf{G}(\theta_{K-1}(n)) \mathbf{G}(\theta_{K-2}(n)) \cdots \mathbf{G}(\theta_1(n)) \mathbf{G}(\theta_0(n))
\]

\[(23)\]

where \(\theta_0(n)\) satisfies

\[
\tan(\theta_i(n)) = \frac{u_i(n)}{\sqrt{u_i^2(n) + v_i^2(n) - 1}} \tag{24}\]

for \(i = 0, 1, 2, \ldots, (K - 1)\).

In Table 1, QRD-RLS based on complex Givens rotations is summarized.

<table>
<thead>
<tr>
<th>Initialization:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathbf{R}[0] = \sqrt{\delta} \mathbf{I}_M \in \mathbb{R}^{M \times M}) with (0 \leq \delta \ll 1), (\mathbf{u}[0] = 0_M)</td>
</tr>
<tr>
<td>for (t = 1, 2, \ldots) do:</td>
</tr>
<tr>
<td>(\mathbf{Q}[t] \begin{pmatrix} \sqrt{\beta} \mathbf{R}[t-1] \ \mathbf{x}[t] \ \mathbf{y}[t] \end{pmatrix} = \begin{pmatrix} \mathbf{R}[t] \ \mathbf{u}[t] \ \mathbf{e}[t] \end{pmatrix})</td>
</tr>
<tr>
<td>where (\mathbf{Q}[t] = \mathbf{Q}_M[t] \cdots \mathbf{Q}_1[t] \in \mathbb{C}^{(M+1) \times (M+1)})</td>
</tr>
</tbody>
</table>
| with \(\mathbf{Q}_m[t] = \begin{pmatrix} \mathbf{I}_{m-1} & : & : \\
: & \cos \theta_m[t] & \sin \theta_m[t] \exp(-j\phi_m[t]) \\
\vdots & \vdots & \vdots \\
\cos \theta_m[t] \exp(-j\phi_m[t]) & \sin \theta_m[t] & \mathbf{I}_{M-m} \end{pmatrix}\) |
| \(\mathbf{e}[t] = \mathbf{e}[t] \cdot \hat{\mathbf{e}}[t] \) with \(\hat{\mathbf{e}}[t] = [\mathbf{I}_{m-1}^M \begin{pmatrix} \cos \theta_m[t] \exp(-j\phi_m[t]) \\
\vdots \\
\cos \theta_m[t] \exp(-j\phi_m[t]) \end{pmatrix}\) |

Table 1: QRD-RLS based on complex Givens rotations
In Fig.1, a three-tap QRD-RLS minimization filter using the QRD-RLS algorithm is presented in its signal flowgraph representation. Each cell in the filter can be implemented with one compound CORDIC processor. The internal cells execute the rotation mode and boundary cell conduct the vectoring mode.

III. Implementation of Given rotations employing CORDIC

The conventional CORDIC algorithm is well suited to carry out the Givens rotation computation [5]. But in original CORDIC algorithm operates on real number system, but it need to handle the complex signal in beamforming application. Thus, in this report we only concern about the modification of CORDIC in order to operate Givens rotations on complex signal. The complex Givens rotations is defined as
It consists of a phase compensation term and a real given rotation. This form of complex Givens rotation differs from the general case for singular value decomposition (SVD) of complex matrix, because the diagonal elements of $R(n-1)$ are positive real numbers in our case. By mapping this algorithm to the CORDIC implementation. We need to construct a new CORDIC cell for complex Givens rotations. It is obvious that the new CORDIC cell consists of three real CORDIC cells. It is shown in Fig.2. In vectoring mode, the imaginary part of $X$ is annihilated by the $\phi$-CPE and subsequently $|X|$ is annihilated in $\theta$-CPE. Then, the complex Givens rotations are encoded by two sequences of rotation coefficients $\sigma_\phi, \sigma_\theta$. By applying these rotation coefficients to the CORDIC cell configured to operate in rotation mode, we can accomplish complex Givens rotations simultaneously.

![CORDIC cell for complex Givens rotation](image)

**IV. Annihilation-reordering Look-ahead Transformation and**
the Systolic Array for Look-ahead Transformed QRD-RLS

Adaptive Filters

A. Principle of this transformation
As indicated in [6], the block data processing by repeatedly applying the QRD-RLS algorithm can be illustrated in Fig. 3. The comparison of the SFGs for sequential and block data processing is shown in Fig. 4(a). In Fig. 4(b), the number of delay elements inside the feedback loop increases to four, but the delay of the feedback loop also increases to four CORDIC processor delays. It is obvious that both SFGs in Fig. 4 have the same iteration bound and thus the same throughput. Therefore, the block data processing only is not sufficient for reducing the iteration bound.

It can be reduced iteration bound by changing the order of the annihilation in Fig. 3 to the order in Fig. 5, a different SFG can be obtained as shown in Fig. 6. Compared to the SFG in Fig. 4(b), the iteration bound in Fig. 6(b) is reduced to one-quarter of the original value.

Fig. 3. Block processing with direct use of the standard QRD-RLS algorithm

Fig. 5. (a) Sequential QR update. (b) Block-processing QR update with block size of four.
B. The Systolic Array for Look-ahead Transformed QRD-RLS Adaptive Filters

With the annihilation-reordering look-ahead transformation for a look-ahead factor of three, the three-tap QRD-RLS minimization filter in Fig. 1 is transformed into new filter architecture, as shown in Fig. 7. In these look-ahead transformed filter architectures, the iteration bounds are reduced to one-third of the original values. However, each cell in Figs. 1 corresponds to three cells in Figs. 7. It is obvious that with a look-ahead...
transformation for a look-ahead factor of M, each cell in Figs. 1 will correspond to M cells in the transformed filter architecture and the iteration bound will be reduced to 1/M of the original value. Therefore, when the look-ahead factor is large, the number of cells and the complexity to perform pipelining or folding transformations will increase dramatically. After look-ahead transformation, we must retime this QRD-RLS adaptive filter in order to minimize the critical path. It can be shown that the critical path can be reduced to iteration bound by retiming the lumped delay element at output. This procedure is shown in Fig. 8.

Fig. 8. SFG of look-ahead transformed QRD-RLS minimization filter.

Fig. 7. SFG of look-ahead transformed QRD-RLS minimization filter

Fig. 10. Retiming internal compound CORDIC processor.

Fig. 8. Retiming internal compound CORDIC processor
V. Fixed point analysis of QRD-RLS adaptive filter

A. Dynamic range of QRD-RLS adaptive filter:
In this section, we derive some of the dynamic range of QRD-RLS. Note that the results obtained here use the original notation of $\hat{\epsilon}$. These are used in the main text by converting to the effective forgetting factor notation. According to the quasi-steady model [7], the content of boundary cells in the QRD-RLS systolic array tend to reach a steady-state value regardless of input data if $\hat{\epsilon}$ is close to unity. Consider the boundary cell equation

$$r(n)^2 = \lambda r(n-1)^2 + u(n)^2.$$  

At steady state, the above equation can be written as (after taking expectation)

$$r_\infty^2 = \lambda r_\infty^2 + E[u(n)^2].$$

For the first boundary cell, the input $u(n)$ is the data input. We finally obtain

$$r_\infty^2(\text{QRD}) = \frac{\sigma_u^2}{1 - \lambda}.$$  

In addition, it can be shown that the dynamic range of the internal cell except the last column in QRD-RLS is smaller than the first boundary cell.

Next, we consider the derivation of average content of internal cells in the last column in QRD-RLS algorithm. It can be found in [8] that dynamic range of internal cell in the last column of QRD-RLS is

$$h_\infty^2(\text{QRD}) = \frac{1}{M(1 - \lambda)}.$$  

Where $M$ is the tap length of filter.

B. Quantization effect of QRD-RLS adaptive filter:
Now, we consider the deviation due to fixed-point arithmetic in the QRD-RLS algorithm. Assume we treat the quantization error as white noise with variance $\sigma_q^2 = 2^{-2k}/12$. According to the recent publishes [9], the deviation in final estimation error can be approximately expressed as
\[ E[\{e(n) - e(n)\}] \approx 0 \quad (3.37) \]

and

\[ E[\{e(n) - e(n)\}]^2 \approx \sigma_u^2 \Delta^{M-1} \left\{ \Delta^M \left( 1 + \frac{1}{M \sigma_u^2} \right)^M - 1 \right\} \cdot \left( 5M + \frac{1.5 \Delta}{1 - \Delta} \right) \sigma_c^2. \quad (3.38) \]

Where \( \sigma_u^2 \) is the energy of inputs, \( M \) is tap length of filter, \( \bar{\epsilon} \) is forgetting factor. \( \sigma_c^2 \) is the variance of quantization error.

In Fig. 9, the deviation values are plotted as function of the number of bits used for the fractional part \( k \).

![Fig.9 Average deviation of estimation error in fixed point arithmetic implementation](image)

VI. Matlab simulation result of QRD-RLS Adaptive filter

In this section, a 5-tap MVDR-QR-beamformer (minimum variance distortionless reponse) based on CORDIC is simulated. In simulation I, we will compare the performances between MVDR-QR without look-ahead and MVDR-QR with look-ahead. In simulation II, we will examine the MVDR-QR in different interference-to-noise ratio. In
In simulation III, we will apply different number of snapshot to our beamformer and evaluate its performance.

**The experiment environments:**
The directions of the target and source of interference are as follows:

<table>
<thead>
<tr>
<th>Excitation</th>
<th>Angle of incidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Arccos(0.2)</td>
</tr>
<tr>
<td>Interference</td>
<td>Arccos(-0.4)</td>
</tr>
</tbody>
</table>

Target-to-noise ratio is 10db.
The CORDIC processor employs 8 macro rotation.
Note: All simulation in this section except simulation IV are applied floating point arithmetic.
Target is rectangle wave with unity amplitude.
Interference and noise are AWGN.

![Figure 10: Input and interference in beamformer](image)

**A. Simulation I:**
In this simulation, each Cordic cell is implemented by conventional Cordic. The look-ahead factor of MVDR-QR with look-ahead is two. The beam-patterns and output of simulation I are as follows:
Based on above simulation results, we observe that the beam-patterns of MVDR-QR with look-ahead factors=2 and MVDR-QR without look-ahead is almost the same. The output wave of MVDR-QR with look-ahead is slightly better than MVDR-QR without look-ahead.

**B. Simulation II:**

The second simulation is to examine the performance of MVDR-QR beamformer with look-ahead factor=2 in different Interference to noise ratio (INR).

The beam-patterns of simulation II are as follows:
The three curves correspond to INR=20db, 30db, 40db. Based on these results from simulation II, the beamformer generates a sharper nulling response in higher INR environment than in lower INR environment.

C. Simulation III:

The third simulation is to examine the performance of MVDR-QR with look-ahead factor=2 in different number of snapshot. The beam-patterns of simulation III are as follows (n is number of snapshot):

From above results, the response of beamformer is relatively insensitive to variations in number of snapshot.
D. Simulation IV:

In this simulation, we examine the performance of QRD-RLS beamformer when fixed-point arithmetic is used. All quantities in the algorithm are represented with $k_i$ bits for the integer part and $k$ bits for fractional part (i.e., $k_i$ bits before the binary point and $k$ bit after binary point) $k_i$ must be set to avoid any overflow. For this simulation, we choose the forgetting factor $\bar{\alpha}=0.99$. Thus, the dynamic range of boundary cell is 10. Because desired signal in MVDR RLS is zero, the dynamic range of all internal cells is smaller than dynamic range of boundary cell. In this simulation, we choose $k=8$, $k=10$, and $k=12$, and observe their performance. The simulation results are as follows:

![The beampatterns in different fixed-point arithmetic implementation](image)

Fig. 13 The beampatterns in different fixed-point arithmetic implementation
Fig. 14 The output waveform of MVDR beamformer in different fixed point arithmetic implementation

Fig. 15 Estimation error of different fixed-point implementation

<table>
<thead>
<tr>
<th>k</th>
<th>Deviation of estimation error (db)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-32.7487</td>
</tr>
<tr>
<td>10</td>
<td>-42.3517</td>
</tr>
<tr>
<td>12</td>
<td>-48.4243</td>
</tr>
</tbody>
</table>

Fig. 16 Deviation of estimation error in different fixed-point implementation
According to our simulated beampatterns, the beampattern of beamformer with k=12 is close to the floating point implementation of QRD-RLS beamformer. Moreover, due to quantization error the gain of target angle in the beampattern of fixed-point implementation is less than unity. This phenomenon causes the degradation of notching performance. In Fig 15 and Fig 16, it can be found that the estimation error converges faster when the word-length is increased. And the deviation of estimation error is close to the theoretical result that we mention before.

VII. Conclusion:

The QRD-RLS adaptive filters will become attractive in radar, sonar, and mobile/wireless communication systems due to their superior convergence rate and increasing chip capacity. However, it is still a challenge to design an ASIC chip architecture for such applications because their systolic array implementations are of large size. Because QRD-RLS exists a recursive path in signal flow diagram, its sampling rate is bounded by this feedback loop. By employing anihilation reordering look-ahead method, we can increase the sampling rate of QRD-RLS. In this paper, a novel low-complexity hierarchical pipelining and folding approach toward ASIC chip designs is presented. In this report, we apply this algorithm to MVDR-beamformer, and verify this look-ahead QRD-RLS in Matlab. Some further performance comparisons is also presented. Moreover, the fixed-point implementation of QRD-RLS MVDR beamformer is also analyzed. The results provide a guide to design a fixed-point implementation of QRD-RLS adaptive filter.

Reference:

