Complex QR Decomposition Using Fast Plane Rotations for MIMO Applications

Aditya Awasthi, *Member, IEEE*, Rohit Guttal, Naofal Al-Dhahir, *Fellow, IEEE*, and Poras T. Balsara, *Fellow, IEEE*

***Abstract*—QR decomposition (QRD) is widely used in various engineering applications and its implementation has a significant impact on the system performance and complexity. This paper develops a low-complexity QRD algorithm based on fast plane rotations, which does not require square-root operations for decomposing a complex-valued matrix. Furthermore, an update-based implementation is presented where computations are performed incrementally as the data arrives sequentially in time to drastically reduce the overall latency and hardware resources. Practical results for QRD-based spatial correlation estimator are provided to demonstrate the effectiveness of our solution for multiple-input multiple-output (MIMO) systems with complex-valued signals.**

***Index Terms*—Least-squares problems, fast plane rotations, square-root free QR-decomposition.**

I. Introduction

MIMO techniques have been widely adopted in wireless and wireline communications because they offer significant improvements in the data transmission rate and quality of services (QoS). QR decomposition (QRD) is a key MIMO technique for decomposing a matrix into the product of an orthonormal matrix and a triangular matrix and is widely used for robust numerical computations of the eigenvalues of a matrix and finding least squares approximations. In recent years, there has been considerable interest in designing QRD algorithms and their associated hardware architectures based on Householder reflections [1], Givens rotations [2] and the Gram-Schmidt procedure [3].

However, the recent literature on QRD implementation mainly focuses on wireless MIMO systems where QRD is typically performed on a large number of small matrices (e.g., decomposition of channel matrices with size up to 8×8). In such systems, the input data for QRD computations i.e., all the elements of a given matrix, is assumed to be available simultaneously. QRD implementation throughput is often improved by pipelining the various row and column operations where each pipeline stage can work on an independent set of inputs [4], [5]. On the other hand, our focus in this paper is on applications involving "overdetermined systems" e.g., computing alien noise correlation in vectored VDSL2 systems [6], [7]. In such applications, the input to the QRD hardware $\mathbf{A} \in \mathbb{C}^{m \times n}$, is provided sequentially over time where m and n correspond to the number of observations (received symbols) and the number of receivers, respectively. Furthermore, the estimation accuracy required often necessitates processing a large number of observations ($m \gg n$) which makes the triangularization of large complex-valued MIMO matrices very challenging due to high computational complexity and large memory requirements. Most of the QRD approaches reported in the literature based on Householder, Givens or Gram-Schmidt procedures perform operations across multiple rows and columns and often involve square roots and multiple division operations. This can make the QRD of large matrices prohibitively complex with impractical latency and memory requirements, since the data received at multiple time instances has to be stored before QRD computations can begin.

In this paper, we make the following contributions. First, we derive the complex-version of the fast plane rotation algorithm from its real counterpart [8], [9] and show that the square-root operations can be avoided and a lower implementation complexity can be achieved for the decomposition of complex-valued matrices as well. Fast plane rotations (also known as "Fast-Givens transforms") were formulated by Gentleman [10] to reduce the number of scalar-vector multiplications and eliminate square roots from the calculation of the plane rotations. Anda and Park [9] later developed fast plane rotations with dynamic scaling to prevent the problem of underflow and overflow, which eliminated the computational overhead of monitoring and periodic rescaling necessitated by the standard fast rotations. Second, we present an iterative QRD approach which exploits the sequential arrival of input data to break up the $O(mn^2)$ QRD computational complexity into m update-steps of $O(n^2)$ complexity and, hence, minimize the overall latency, silicon area and memory required for practical implementation. Third, we quantify the benefits of our proposed approach by considering the problem of alien noise mitigation for upstream vectored-VDSL2 transmission as a case study to demonstrate the effectiveness of our approach for MIMO systems with complex-valued signals. The following notation is used in the rest of this paper. Bold uppercase letters indicate matrices. The $\|.\|$ indicates Euclidean norm, and the superscripts T and H denote the transpose and the complex conjugate-transpose (Hermitian) operations, respectively.

II. QR Decomposition Approach Using Complex Fast Plane Rotation

Plane rotations are widely used in various algorithmic contexts, e.g., the QR decomposition, singular value decomposition, and reduction to the Hessenberg form. Givens rotation $\mathbf{G}_n(i, k, \theta)$ of order n through an angle θ in (i, k) plane is a

Manuscript received January 6, 2014; revised July 7, 2014; accepted August 5, 2014. Date of publication August 21, 2014; date of current version October 8, 2014. This research work was carried out at Xtendwave, 7920 Belt Line Road, Suite 1000, Dallas, TX 75254 USA under NSF SBIR Award #1152622. The associate editor coordinating the review of this paper and approving it for publication was M. Matthaiou.

A. Awasthi and R. Guttal are with Xtendwave, Dallas, TX 75254 USA (e-mail: aditya.awasthi@gmail.com).

N. Al-Dhahir and P. T. Balsara are with the Department of Electrical Engineering, University of Texas at Dallas, Richardson, TX 75080 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LCOMM.2014.2349995

rank-two correction to the identity matrix [8], which involves computation of cosine and sine terms (refer to Table II for details). For a given $x = [f\ g]^T \in \mathbb{C}^2$ and $d = [d_f\ d_g]^T \in \mathbb{R}_+^2$, the fast plane rotation (also referred as the complex Fast-Givens transformation) $\mathbf{M}$ is computed using Algorithm 1 such that the second component of $\mathbf{M}^H x$ is zero and $\mathbf{M}^H\mathbf{D}\mathbf{M}$ is a diagonal matrix, as shown below

$$\mathbf{M} = \underbrace{\begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix}}_{type=1} \text{ or } \underbrace{\begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix}}_{type=2}$$

$$\mathbf{M}^H x = \begin{bmatrix} r \\ 0 \end{bmatrix} \ \& \ \mathbf{M}^H \underbrace{\begin{bmatrix} d_f & 0 \\ 0 & d_g \end{bmatrix}}_{=\mathbf{D}} \mathbf{M} = \underbrace{\begin{bmatrix} d_f^{new} & 0 \\ 0 & d_g^{new} \end{bmatrix}}_{=\mathbf{D}^{new}} \quad (1)$$

where, α, β, r, $type$, d_f^{new}, and d_g^{new} are the outputs of the function *fast.givens* of Algorithm 1. The key idea behind QR decomposition using fast plane rotations is that the square roots needed for the computation of the cosine and sine can be eliminated. Consequently, the number of multiplications can be reduced through the factorization $\mathbf{A} = \mathbf{R}\mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix and $\mathbf{R}$ is accordingly scaled. During QRD, these two factors are updated separately and their product is only calculated at the end, if an explicit result is required.

Algorithm 1: Complex Fast-Givens Transform

$[\alpha, \beta, r, type, d_f^{new}, d_g^{new}] = \mathbf{fast.givens}(f, g, d_f, d_g)$
if $f = 0$ **then**
  $type = 1$; $\alpha = \beta = 0$; $r = g$;
  $d_f^{new} = d_g$; $d_g^{new} = d_f$;
else if $g = 0$ **then**
  $type = 2$; $\alpha = \beta = 0$; $r = f$;
  $d_f^{new} = d_f$; $d_g^{new} = d_g$;
else if $\|f\|^2 \leq \|g\|^2$ **then**
  $type = 1$ $i = f/g$; $s = d_g/d_f$;
  $\alpha = -\bar{i}$; $\beta = s * i$;
  $\gamma = s * \|i\|^2$; $r = g * (1 + \gamma)$;
  $d_f^{new} = (1 + \gamma) * d_g$; $d_g^{new} = (1 + \gamma) * d_f$;
else
  $type = 2$; $i = g/f$; $s = d_f/d_g$;
  $\alpha = -\bar{i}$; $\beta = s * i$;
  $\gamma = s * \|i\|^2$; $r = f * (1 + \gamma)$;
  $d_f^{new} = (1 + \gamma) * d_f$; $d_g^{new} = (1 + \gamma) * d_g$;
end

The Fast-Givens QRD using complex Fast-Givens transformations is shown in Algorithm 2. The "full QRD" algorithm (Algorithm 2A) initializes the diagonal matrix $\mathbf{D}$ as the identity matrix and $\mathbf{R}_{full}$ as the input matrix $\mathbf{A}$. The elements of matrix $\mathbf{R}_{full}$ are updated iteratively by overwriting the previous values to ultimately arrive at the final result of a triangular matrix $\mathbf{R}_{full}$. An appropriate form of Fast-Givens transformation is selected for each iteration in Algorithm 2A (based on the calculated "$type$") in order to minimize the growth factor $(1 + \gamma)$ of the entries in $\mathbf{R}_{full}$ and $\mathbf{D}$. Overflow can be effectively managed by using dynamic scaling techniques as described in [9]. The complex Fast-Givens QRD can also be broken into incremental QRD-update steps as shown in Algorithm 2B to exploit the sequential arrival of input data. During each incremental QRD-update step, the incoming input data row-vector is stored in a FIFO buffer (vector **new** as shown in line 9 of Algorithm 2B). Next, complex Fast-Givens transformations are used to zero-out the input data row-vector elements (inner for-loop on line 11) and to update upper-triangular matrix $\mathbf{R}$. Once the upper triangular matrix $\mathbf{R}$ has been updated, the next set of inputs (next observation) are written into the FIFO buffer (i.e., **new** vector is overwritten), and the QRD-update process is repeated until all of the input data to be used for QRD is exhausted.

Algorithm 2: Complex Fast-Givens QRD algorithm

Input: $\mathbf{A} \in \mathbb{C}^{m \times n}$, $m \geq n$
Output: $\mathbf{R}_{full} \in \mathbb{C}^{m \times n}$, where last $(m - n)$ rows are zero i.e. $\mathbf{R}_{full} = [\mathbf{R}; 0^{(m-n)\times n}]$, $\mathbf{R} \in \mathbb{C}^{n \times n}$

A. Full QRD algorithm:

1 **Initialize**: $\mathbf{R}_{full} = \mathbf{A}$, $\mathbf{D} = \mathbf{I}_m$ (Note: $d \equiv trace(\mathbf{D}) = [11...1]^T$)
2 **for** $j = 1 : n$
3 **for** $i = m : -1 : j + 1$
  Step 1: **Get α and β using Algorithm 1**
  $[\alpha, \beta, \mathbf{R}_{full}(i\text{-}1,j), type, d(i\text{-}1), d(i)] =$
  fast.givens($\mathbf{R}_{full}(i\text{-}1,j)$, $\mathbf{R}_{full}(i,j)$, $d(i\text{-}1)$, $d(i)$);
  $\mathbf{R}_{full}(i, j) = 0$;
  Step 2: **Update elements based on** $type$
  if $j < n$ and $type = 1$ **then**
   $\mathbf{R}_{full}(i\text{-}1{:}i,j{+}1{:}n) = \begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix}^H \mathbf{R}_{full}(i\text{-}1{:}i,j{+}1{:}n)$;
  else if $j < n$ and $type = 2$ **then**
   $\mathbf{R}_{full}(i\text{-}1{:}i,j{+}1{:}n) = \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix}^H \mathbf{R}_{full}(i\text{-}1{:}i,j{+}1{:}n)$;
  end
4 **end**
5 **end**

B. Update-based Fast-Givens QRD algorithm:

6 **Initialize**: $\mathbf{R} = 0^{n\times n}$, $\mathbf{D} = \mathbf{I}_n$ $(d \equiv trace(\mathbf{D}) = [\underbrace{11...1}_{n}]^T)$
7 $\mathbf{R}(1,1{:}n) = \mathbf{A}(1,1{:}n)$ */*Reading first row*/*
8 **for** $i = 2 : m$ */*Accessing data "row-wise"*/*
9 $\mathbf{new} = \mathbf{A}(i, 1 : n)$; $d_{new} = 1$;
10 **if** $i \leq n$ **then** $k = i$; **else** $k = n + 1$; **end**
11 **for** $j = 1 : k - 1$
  Step 1: **Get α and β using Algorithm 1**
  $[\alpha, \beta, \mathbf{R}(j,j), type, d(j), d_{new}] =$
  fast.givens($\mathbf{R}(j,j)$, $\mathbf{new}(1,j)$, $d(j)$, d_{new});
  Step 2: **Update elements based on** $type$
  if $j < n$ and $type = 1$ **then**
   $\begin{bmatrix} \mathbf{R}(j,j{+}1{:}n) \\ \mathbf{new}(1,j{+}1{:}n) \end{bmatrix} = \begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix}^H \begin{bmatrix} \mathbf{R}(j,j{+}1{:}n) \\ \mathbf{new}(1,j{+}1{:}n) \end{bmatrix}$;
  else if $j < n$ and $type = 2$ **then**
   $\begin{bmatrix} \mathbf{R}(j,j{+}1{:}n) \\ \mathbf{new}(1,j{+}1{:}n) \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix}^H \begin{bmatrix} \mathbf{R}(j,j{+}1{:}n) \\ \mathbf{new}(1,j{+}1{:}n) \end{bmatrix}$;
  end
12 **end**
13 **if** $k \leq n$ **then**
14 $\mathbf{R}(k,1{:}n) = \mathbf{new}(1,1{:}n)$; $d(k) = d_{new}$;
15 **end**
16 **end**

Tables I–III compare the computations required by the Householder, Givens and Fast-Givens transformations based complex-valued QRD algorithms, respectively. Note that the complex Fast-Givens QRD does not require any square root operation. Although, the square root operations can also be

TABLE I
HOUSEHOLDER QR COMPLEXITY

Algorithm Steps	Hardware operations			
Input: $\mathbf{A} \in \mathbb{C}^{m\times n}$, $m \geq n$	Note: $\mu = m-k+1$;			
Output: $\mathbf{R} \in \mathbb{C}^{m\times n}$	$\alpha = n-k+1$;			
Initialize: $\mathbf{R} = \mathbf{A}$	**Add**	**Mul**	**Div**	**Sqrt**
for $k = 1:n$				
$\mathbf{x} = \mathbf{R}(k{:}m,k)$;				
$\mathbf{e} = [1\underbrace{0\,0\ldots0}_{m-k}]^T$;				
$\mathbf{u} = \mathbf{sgn}(x(1))\|\mathbf{x}\|\mathbf{e} + \mathbf{x}$;	6μ-4	2μ+2	2	2
$\mathbf{u} = \mathbf{u}/\|\mathbf{u}\|$;	2μ-1	2μ	2μ	1
$\mathbf{R}(k{:}m,k{:}n) =$				
$\mathbf{R}(k{:}m,k{:}n)$	$8\mu\alpha$	$8\mu\alpha$	0	0
$-2(\mathbf{u}\mathbf{u}^H)\mathbf{R}(k{:}m,k{:}n)$;	-2α			
$\mathbf{U}(k{:}m,k) = \mathbf{u}$;				
end				
Total hardware operations				
Add:	$4mn^2 - \frac{4}{3}n^3 + 12mn - 5n^2 - \frac{2}{3}n$			
Mult:	$4mn^2 - \frac{4}{3}n^3 + 8mn - 2n^2 + \frac{16}{3}n$			
Div:	$2mn - n^2 + 3n$			
Sqrt:	$3n$			

TABLE II
GIVENS QR COMPLEXITY

Algorithm Steps	Hardware operations			
Input: $\mathbf{A} \in \mathbb{C}^{m\times n}$, $m \geq n$	Note:			
Output: $\mathbf{R} \in \mathbb{C}^{m\times n}$	$\mu = n-j$;			
Initialize: $\mathbf{R} = \mathbf{A}$	**Add**	**Mul**	**Div**	**Sqrt**
for $j = 1:n$				
for $i = m:-1:j+1$				
$[c, s, \mathbf{R}(i\text{-}1,j)] =$				
givens.rot($\mathbf{R}(i\text{-}1,j)$,$\mathbf{R}(i,j)$));	5	10	5	2
$\mathbf{R}(i,j) = 0$				
if $j < n$				
$\mathbf{R}(i\text{-}1{:}i,j{+}1{:}n) =$				
$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^H \mathbf{R}(i\text{-}1{:}i,j{+}1{:}n)$;	8μ	12μ	0	0
end				
end				
end				
$[c, s, r]$ =***givens.rot***(f,g)				
$c = \|f\|/\sqrt{\|f\|^2 + \|g\|^2}$;				
$s = -sgn(f)\bar{g}/\sqrt{\|f\|^2 + \|g\|^2}$;				
$r = sgn(f)\sqrt{\|f\|^2 + \|g\|^2}$;				
Total hardware operations				
Add:	$4mn^2 - \frac{4}{3}n^3 + mn - \frac{5}{2}n^2 - \frac{7}{6}n$			
Mult:	$6mn^2 - 2n^3 + 4mn - 5n^2 - 3n$			
Div:	$5mn - \frac{5}{2}n^2 - \frac{5}{2}n$			
Sqrt:	$2mn - n^2 - n$			

avoided using the CORDIC algorithm, however, it suffers from high computation latency [5]. The round-off properties of a Fast-Givens transformation are the same as in Givens rotation techniques [8], [9]. The update-based Fast-Givens QRD (Algorithm 2B) divides the QRD computations into m update-steps of $O(n^2)$ complexity, where the update-step computations can begin as soon as new input data (input matrix row) is received instead of waiting for the entire input data matrix. If the computations for the update-step are completed before the arrival of the next set of data, the processed input data row-vector can be discarded from memory (input FIFO) to make space for new incoming data. Thus, the overall latency and memory requirements can be drastically reduced.

TABLE III
FAST-GIVENS QR COMPLEXITY

Algorithm Steps	Hardware operations			
Note: $\mu = n-j$;	**Add**	**Mul**	**Div**	**Sqrt**
For **Step1** of **Algorithm 2A**, or **Step1** of **Algorithm 2B**	6	17	3	0
For **Step2** of **Algorithm 2A**, or **Step2** of **Algorithm 2B**	8μ	8μ	0	0
Total hardware operations for Algorithm 2A				
Add:	$4mn^2 - \frac{4}{3}n^3 + 2mn - 3n^2 - \frac{5}{3}n$			
Mult:	$4mn^2 - \frac{4}{3}n^3 + 13mn - \frac{17}{2}n^2 - \frac{43}{6}n$			
Div:	$3mn - \frac{3}{2}n^2 - \frac{3}{2}n$			
Sqrt:	0			
Hardware operations per data-row for Algorithm 2B				
Add:	$4n^2 + 2n$			
Mult:	$4n^2 + 13n$			
Div:	$3n$			
Sqrt:	0			

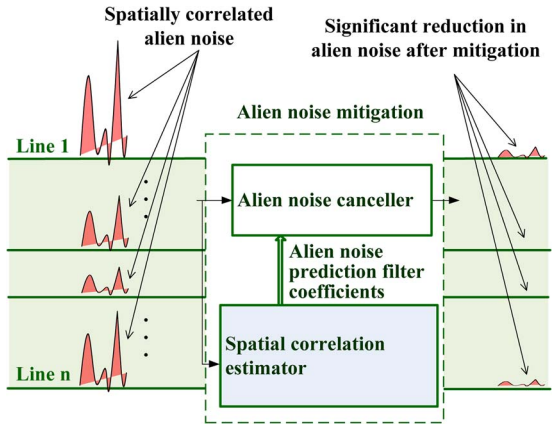

Fig. 1. QR decomposition based alien noise mitigation.

III. CASE STUDY: ALIEN NOISE MITIGATION

This section compares the performance of our proposed Fast-Givens "full QRD" and "update-based QRD" (Algorithms 2A & 2B). We have considered the problem of alien noise mitigation in vectored-VDSL2 transmission [7], which involves QRD of a large complex-valued alien noise matrix $\mathbf{A} \in \mathbb{C}^{m\times n}(m \gg n)$, for estimating the alien noise spatial correlation across the n vectored-VDSL2 lines (receivers) based on alien noise observed at each line during m training symbols, as shown in Fig. 1. The alien noise prediction filter coefficients for each frequency subcarrier are given by the elements of the normalized upper-triangular matrix whose jth row $\tilde{\mathbf{R}}(j,:) = \mathbf{R}(j,:)/\mathbf{R}(j,j)$, where $\mathbf{R}$ is obtained from QRD of input noise matrix $\mathbf{A} = \mathbf{QR}$.

Floating point computation of alien noise prediction filter coefficients, $\tilde{\mathbf{R}}$ using our proposed full and update-based Fast-Givens QRD exactly matches the result obtained by any other QRD method (Householder, Givens or Gram-Schmidt). The normalized error (= *finite-precision value* − *floating-point value*)/*floating-point value*) for different wordlength is shown in Fig. 2 for QRD implementations based on our proposed full

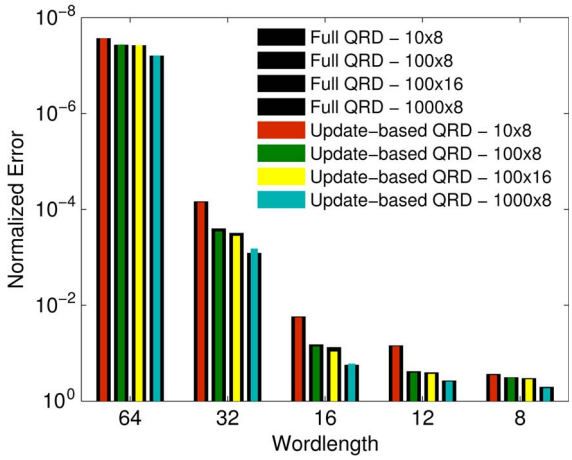

Fig. 2. Error in full and update-based Fast-Givens QRD.

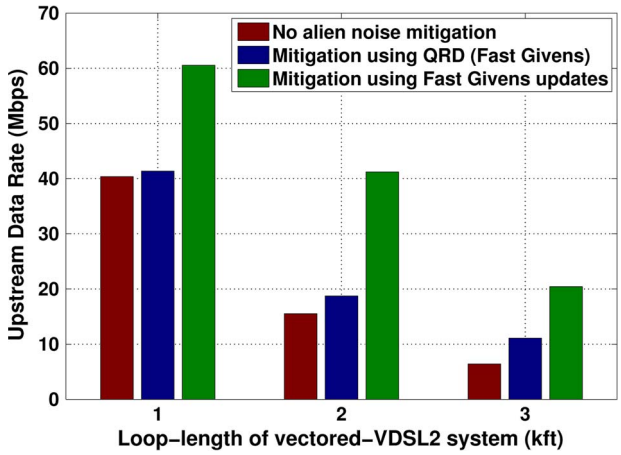

Fig. 3. Data rate improvement using QRD-based alien noise mitigation.

and update-based Fast-Givens QRD. For Fast-Givens QRD to work with any wordlength, appropriate scaling by power of 2 (*hardware shift*) are necessary at two stages

1) Fast-Givens QRD uses real-valued inversions (e.g., $1/d_f$, $1/\|f\|^2$ etc.). The input to the inversion block must be scaled to be within a range for minimizing error (range is [1,8) for this paper).
2) The f and g inputs of Fast-Givens transformation block (Algorithm 1) are scaled by the same factor such that the magnitudes of $re(f), im(f), re(g), im(g)$ is less than 1. Similarly, the d_f and d_g inputs are scaled by another factor to be within [1,8).

As seen in Fig. 2, the normalized error for full and update-based Fast-Givens QRD is almost the same and the error for both increases as wordlength decreases. The performance also degrades with increasing input matrix dimension since elements in $\tilde{\mathbf{R}}$ grow by $1+\gamma$ (see Algorithm 1) and processing more elements increases the dynamic range. Fig. 2 assumes unlimited hardware and computation time for both full and update-based Fast-Givens QRD. However, if we consider a typical residential vectored VDSL2 deployment [11] (300 lines, VDSL2 profile-17a [12]), the memory required to store each input data row is about 1.4 megabyte (MB) assuming 16-bit data-wordlength for real and imaginary noise signals. Since update-based Fast-Givens QRD breaks the $O(mn^2)$ complexity of the full Fast-Givens QRD into m update-steps of $O(n^2)$ complexity, which can begin as soon as each input data-row (sync-symbol) arrives, memory requirements are drastically reduced and a much larger number of of VDSL2 training symbols can be processed to improve the estimation accuracy of alien noise prediction filter coefficients. Fig. 3 shows the system performance for full and update-based Fast-Givens QRD assuming total computational time of 0.54 ms. Hardware for the spatial correlator block capable of handling up to 300 VDSL2 lines consists of $4k$ multipliers, $7.2k$ adders, $6k$ registers and $7k$ multiplexers. Total gate count and the clock frequency for the spatial correlator block was 7.8 M gates and 200 MHz, respectively, using IBM 65 nm standard cell library.

IV. Conclusion

We developed a QRD algorithm based on complex fast plane rotations, which does not require square-root operations. An update-based implementation of the QRD algorithm which exploits the sequential arrival of input data was described to drastically reduce the implementation complexity and latency. Taking vectored-VDSL2 system as a case study, complexity comparisons, simulation results and gate counts were presented to illustrate the significant gain in system performance and reduction in implementation complexity.

References

[1] I. Kurniawan, J. Yoon, and J. Park, "Multidimensional householder based high-speed QR decomposition architecture for MIMO receivers," in *Proc. IEEE ISCAS*, 2013, pp. 2159–2162.
[2] Z. Huang and P. Tsai, "Efficient implementation of QR decomposition for gigabit MIMO-OFDM systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 10, pp. 2531–2542, Oct. 2011.
[3] R.-H. Chang, C.-H. Lin, K.-H. Lin, C.-L. Huang, and F.-C. Chen, "Iterative QR decomposition architecture using the modified Gram-Schmidt algorithm for MIMO systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 5, pp. 1095–1102, May 2010.
[4] P. Luethi *et al.*, "VLSI implementation of a high-speed iterative sorted MMSE QR decomposition," in *Proc. IEEE ISCAS*, May 2007, pp. 1421–1424.
[5] P. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.
[6] G. Ginis and J. Cioffi, "Vectored transmission for digital subscriber line systems," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 5, pp. 1085–1104, Jun. 2002.
[7] A. Awasthi, N. Al-Dhahir, O. Eliezer, and P. Balsara, "Alien crosstalk mitigation in vectored DSL systems for backhaul applications," in *Proc. IEEE ICC*, Jun. 2012, pp. 3852–3856.
[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The John Hopkins Univ. Press, 1996.
[9] A. A. Anda and H. Park, "Fast plane rotations with dynamic scaling," *SIAM J. Matrix Anal. Appl.*, vol. 15, no. 1, pp. 162–174, Jan. 1994.
[10] W. M. Gentleman, "Least squares computations by Givens transformations without square roots," *IMA J. Appl. Math.*, vol. 12, no. 3, pp. 329–336, Dec. 1973.
[11] V. Oksman *et al.*, "The ITU-T's new G.vector standard proliferates 100 Mbps DSL," *IEEE Commun. Mag.*, vol. 48, no. 10, pp. 140–148, Oct. 2010.
[12] *Very high speed digital subscriber line transceivers 2 (VDSL2)*, ITU-T Recommendation G.993.2, Feb. 2006.