Received August 2, 2020, accepted August 12, 2020, date of publication August 18, 2020, date of current version September 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017558

Superior Implementation of Accelerated QR Decomposition for Ultrasound Imaging

S. G. SREEJEESH[1,2], **R. SAKTHIVEL**[2], **(Senior Member, IEEE), AND JAYARAJ U. KIDAV**[1]

[1]National Institute of Electronics and Information Technology Calicut, Calicut 673601, India
[2]Vellore Institute of Technology, Vellore 632014, India

Corresponding author: R. Sakthivel (rsakthivel@vit.ac.in)

This work was supported in part by the Medical Electronics and Health Informatics (ME and HI) Division of the Ministry of Electronics and Information Technology, Government of India, through the ICDU Project, under Grant 1(5)/2015-ME and HI dated 18/11/2015, and in part by the Microelectronics Division of the Ministry of Electronics and Information Technology, Government of India, through the SMDP-C2SD Project, under Grant 9 (1)/2014-MDD dated 15-12-2014.

ABSTRACT In this work, a novel Minimum Variance Distortion less Response (MVDR) beamformer architecture in which the adaptive weight vector is computed based on modified Column wise Givens Rotation (CGR) is presented. As compared to the conventional MVDR beamformer, Quadrature Rotation Decomposition (QRD)-MVDR is suitable for hardware realizations. To improve the real-time performance requirements of the MVDR Beamformer, a parallel pipe-lined CGR based QRD architecture is employed in the adaptive weight computation stage of the MVDR Beamformer. A computationally efficient MVDR beamforming algorithm, which requires to compute only the R matrix in QRD, rather than matrix inverse is used to develop the architecture. The developed architecture generates the adaptive weight vector in 3.9ns, and hence a beam can be formed in 0.25msec time. The designed architecture is implemented using Verilog Register Transfer Level (RTL) coding, and the functional equivalence checking was carried with the Verasonics Vantage-64 Ultrasound Research Platform (URP). The architecture is also ported on Xilinx Kintex-7 FPGA based Emulation set up and validated in real-time, targeting medical ultrasound imaging applications. The developed architecture is compared with the existing architecture implementations. It concludes that the architecture is superior in terms of computational time and can be adapted for ultrafast adaptive beamforming applications.

INDEX TERMS Adaptive beamforming, accelerated MV-beamformer, MVDR, adaptive weight computation, FPGA prototype, medical ultrasound, CWGR, parallel architecture.

I. INTRODUCTION

Delay-and-sum (DAS) is a very fundamental technique for beamforming, given a known direction of arrival (DOA) and time delays. It sums the delayed signal depending on the direction of arrival of the signal. Here the system produces destructive interference in all directions but the not in the direction of arrival/interest [1]. This beamformer combines the signal from different channels with fixed and predefined weights. It is the simplest beamformer for computational complexity, low resolution, and reduced interference suppression makes it the worst one for medical imaging. Adaptive beamformers use adaptive weights, which depend on the data-adaptive apodization weights. Minimum Variance Distortionless Response(MVDR) is such an algorithm that uses adaptive weights. This algorithm is far better in resolution when compared to DAS, the expense of which is a very high computational complexity [2]. Figure 1 shows the conventional Delay and Sum Beamformer structure. The achieved resolution improvement is due to the suppression of interference from undesired direction while preserving the signal from the desired direction. The most expensive/high computational block is the one that calculates the spatial covariance matrix and its inverse for the computation of adaptive weights. MVDR Beamforming improves the resolution of the image to a large extent compared to the DAS Method [3]. The implementation issues associated with MVDR is the calculations involved in the algorithm. QR Decomposition reduces the computational complexity of the MVDR. This method does not require the computation of the spatial covariance matrix and the inverse of it, all this at the expense of necessary calculations for the QR Decomposition.

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang.

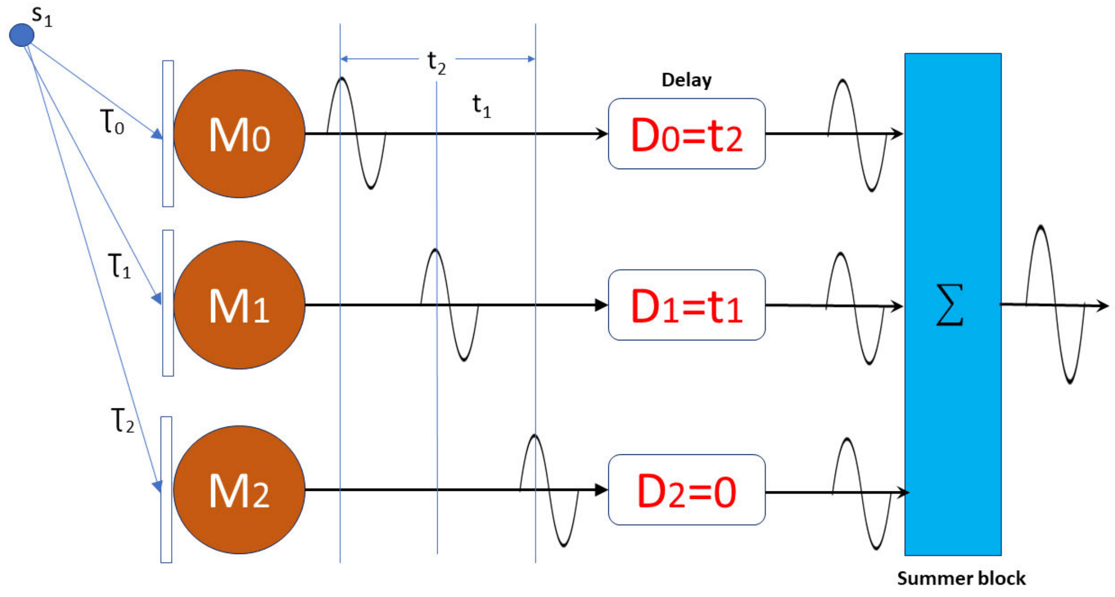


FIGURE 1. Delay and sum beamformer-DAS.

A modified algorithm that further reduces the computational complexity, which is accelerated QR-MV beamformer, as this method only requires R calculation. Many methods are available for R calculations, like GR, MSGR, and Fast Toeplitz orthogonalization, etc. Column Wise Given Rotation is the best-suggested method as it requires very fewer calculations compared to the normal Given Rotation and MSGR methods. The previous articles suggest that only R calculation is required for computation of the adaptive weight vectors for MVDR Beamformer, which takes the major computational complexity for these algorithms. The proposed works use the QR-MV Beamformering techniques to implement a computationally superior method compared to native methods. The major contributions in this article are as follows:

- We present CGR based QRD-MV Beamformer implementation on Verasonics Ultrasound Research Platform (URP) and on Xilinx 7 series (Virtex) FPGA.
- We present a novel modified architecture for CGR based QR Decomposition Algorithm (PCGR-QRD) and MV Beamformer realization using this QRD Algorithm on FPGA Platform.
- We compare our experimental results with the contemporary implementations in the literature and show significant improvement in throughput and resolution at par with native MVDR Beamformer.

The organization of this article is as follows: Section II describes the background of MVDR Beamforming, Section III, the literature survey, Section IV describes the accelerated MV- beamformer implementation using CGR Algorithm, Section V describes the proposed architecture. Section VI describes the experiment setup Section VII deals with the results and discussions, and Section VIII presents the conclusions.

II. BACKGROUND OF MVDR BEAMFORMING

The MVDR is a technique introduced by Capon in 1969. This algorithm can resolve signals which are separated by antenna beam width's fractions [3]. To maximize the signal to noise ratio, this Beamformer takes the desired signal in the (DOA)-direction of arrival. The weight vector of the Beamformer will calculate the desired signal. MDVR Algorithm can maximize the sensitivity of the sensors in only one direction. The Beamformer significantly reduces the Beamformer's output power under a single linear constraint on the array's response to the desired signal [4]. The algorithm depends on the steering vectors, which depend on the angle of incidence of the received signal from the array antenna unit. The direction of the useful signal must be known, and the output power in the direction of the desired signal must be minimized, subject to a unit gain constraint. Figure 2 shows the beamforming concept.

$$0 \leq \mu \leq \frac{1}{2^{*}\lambda max} \tag{1}$$

where λmax is the largest eigen value which is given by the equation 2 of the array correlation matrix R_{xx}(k).

$$Rxx(k) = x(k)X^{H}(k) \tag{2}$$

The received signal vector is denoted x(k). The weights of the array are updated according to the equation given below.

$$w(k+1) = w(k) + \mu e^{*}(k)x(k) \tag{3}$$

the below equation gives the error signal e(k)

$$e(k) = d(k) - w^{H}(k)x(k) \tag{4}$$

To achieve the minimum power of the sensor signals, which are weighted, the Minimum Variance (MV) beamformer continually updates the apodization weights under the

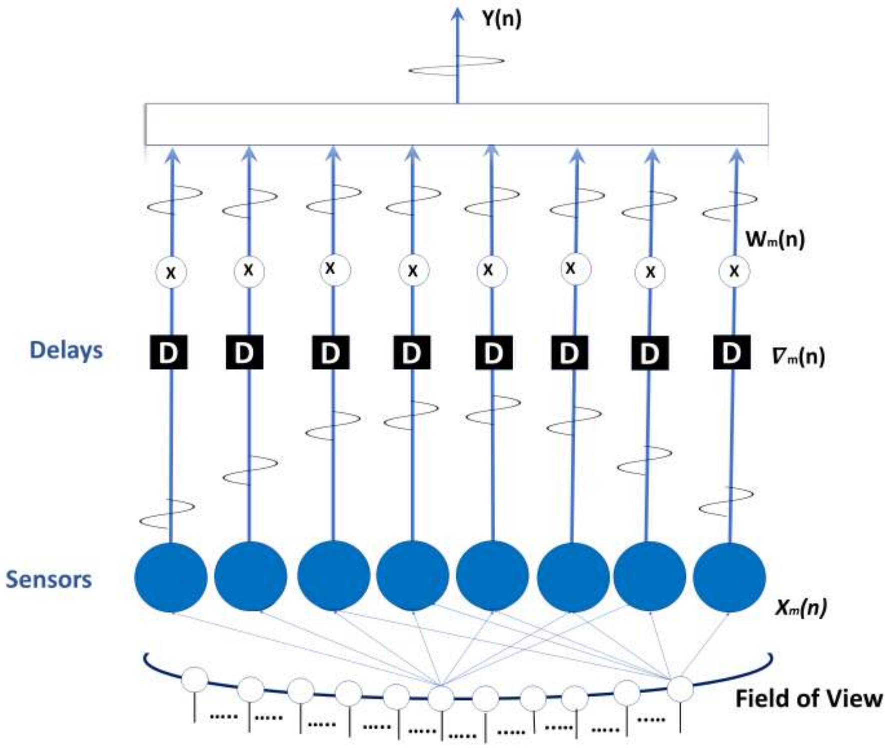


FIGURE 2. **Beamforming diagram.**

restriction of the signal being transmitted without distortion from the point of interest. MVDR Beamformer does not require knowledge of the directions of interference for weight vector calculations. It only needs the direction of the desired signal. The MVDR / Capon beamformer can attain a more prominent resolution than the standard (Bartlett) method, but since it is a full=rank matrix inversion, this algorithm has higher complexity. Technical developments in GPU have started closing this gap and render Capon beamforming possible in real-time. The linearly constrained minimum variance (LCMV) beamformer is the most commonly used criteria in the beamformer for suppressing the interfering signals and noise. The algorithm was first proposed by Frost [4]. Griffiths and Jim [5] made a practical implementation of LCMV by suggesting a specific sidelobe canceller (GSC). Since the Generalized Side Lobe Canceller Algorithm uses an unconstrained approach instead of a constrained algorithm, weights can be modified at much higher rates [6]. Separate unitary transforms, such as Discrete Fourier Transforms (DFTs), have been used to de-correlate input data to improve the convergence rate of the original LMS GSC model [7]. Glentis attempted to reduce the computational complexity of both GSC-LMS and TD-LMS GSC by treating complicated transmissions as a few real signals and using the algorithmic strength reduction method [8], [9]. This discrepancy is enough that the GSC appears to misinterpret the required signal element by nullifying instead of retaining a distortion-free response to it. In reality, almost all beamformer applications lack the required signal during the training period [10-11], which allows the beamformer to be more reliable in the array response to mismatch errors. The beamformed output y[n] at point n in the scanline can be described as

$$y[n] = \sum_{m}^{M-1} w_m[n] x_m[n - \delta_m[n]] \tag{5}$$

where M is the array size, $w_m$is the complex apodization weight, $x_m[n]$ is the sampled data obtained from array element m, and $\delta_m[n]$ is the delay applied to array element m to focus the beam at point n. Simplifying the equation (5) we get

$$y[n] = w^H[n]x[n] \tag{6}$$

where x[n] denotes

$$x[n] = \begin{bmatrix} x_0[n-\delta_0[n]] \\ x_1[n-\delta_1[n]] \\ . \\ . \\ . \\ x_{M-1}[n-\delta_M - 1[n]] \end{bmatrix} \tag{7}$$

w[n] is denotes as

$$w[n] = \begin{bmatrix} w_0^* \\ x_1^* \\ . \\ . \\ . \\ x_{M-1}^* \end{bmatrix} \tag{8}$$

here we denote the complex conjugate by the symbol * and H denotes the rearrangement of a vector or matrix by the complex conjugate.

The MVDR beamformer uses the adaptive apodization weights to minimize the beamformer's output power while maintaining the response from the direction of interest as a constant, which is not maintained in the DAS Beamformer [12].

$$\begin{aligned} E[y^2] &= w^H[n]^* R[n]^* w[n] \;\; for\ min\ w[n] \\ provided\ \ w^H[n]a[n] &= 1 \end{aligned} \tag{9}$$

where E denotes the expectation operator, $R[n] = E[x[n]x[n]^H]$ is the spatial covariance matrix, and a[n] is the steering vector that characterizes the response from the focal point. a[n] is set to [11…1]T because the array data are already time-delayed. This problem can be solved by using the method of Lagrange multipliers. Assuming that R[n] is non-singular, we get a solution as

$$w[n] = \frac{R^{-1}[n]a[n]}{a^H[n]R^{-1}[n]a[n]} \tag{10}$$

Due to coherence, the signal from (DOA) direction of interest and the interfering signals are coherent or highly correlated because all signals which are received are scaled and delayed replicas of the transmitted pulse, which may cause signal cancellation and generate poor beamformed output. This problem will be solved using a subarray averaging technique [12], which makes the MVDR Beamformer a highly computationally complex one, which requires the calculation of R[n] and its inverse. In real-time applications, the covariance matrix is obscure and must be assessed from data. The computational cost included calculating spatial covariance matrix and its inverse limits the usage of MVDR Beamformer for real-time applications like Therapeutic Ultrasound Imaging. Whereas considering the real-time implementation, the complexity of the covariance matrix generation and its inversion become a difficult assignment. In sub-array MVDR, to decrease the size of the spatial covariance matrix, the transducer cluster is partitioned into overlapping subarrays, and the covariance matrices for each sub-array is averaged over the array; comes about producing covariance matrix and its inverse practically implementable.

III. LITERATURE SURVEY

Our conventional MVDR Beamformer has high computational cost as we need to calculate the R[n] and its inverse, which does not allow us to apply this in practical implementations despite its excellent performance. QR Decomposition does not require the R[n] and its inverse calculations, which, when applied on the beamformer, will a better Beamformer with less computational complexity [12]. The transducer array is split into 'P' (P=M-L+1) overlapping sub-arrays, covariance is generated and averaged across the array. This method is termed as spatial smoothing [12].

The subarray data vector is given by

$$x_l[n] = \begin{bmatrix} x_l[n] \\ x_{l+1}[n] \\ . \\ . \\ . \\ x_{l+L-1}[n] \end{bmatrix} \tag{11}$$

here L is the subarray size, which takes values l = 0, 1, 2… M-L. In order to make the subarray vectors into orthonormal vectors, we apply a transformation to make the weight calculation simpler [12]. The subarray data vector in the transform domain is given by

$$z_l[n] = T[n]x_l[n] \tag{12}$$

The steering vector is given by

$$a[n] = [1\ 1\ 1 \ldots\ldots 1]^T \tag{13}$$

The steering vector in transform domain is given by

$$a_z[n] = T[n]a[n] \tag{14}$$

The weight vector in transform domain is given by

$$w_z[n] = \frac{a_z[n]}{a_z^H[n]a_z[n]} \tag{15}$$

The beamformer output [12] is given by

$$y[n] = \frac{1}{M-L+1}\sum_{l=0}^{M-L} W_z^H[n]z_1[n] \tag{16}$$

Transformation matrix T[n] has to be calculated by defining a sub-array data matrix U[n] of size (M-L+1) x L

$$U[n] = \begin{bmatrix} x_{M-L}[n] & x_{M-L+1}[n] & \ldots. & x_{M-1}[n] \\ x_{M-L-1}[n] & x_{M-L}[n] & \ldots. & x_{M-2}[n] \\ .. & .. & .. & \\ .. & .. & .. & \\ .. & .. & .. & \\ x_n[n] & x_1[n] & \ldots. & x_{L-1}[n] \end{bmatrix} \tag{17}$$

QR Decomposition algorithm is applied to the matrix

$$U[n] = Q[n]S[n] \tag{18}$$

where Q[n] is MxN Orthogonal matrix and S[n] is an NxN upper triangular matrix.

$$U^T[n] = Q^T[n]S^T[n] \tag{19}$$
$$Q^T[n] = S^{-T}[n]U^T[n] \tag{20}$$

The transformation matrix can be derived from the above equations as

$$T[n] = S^{-T}[n] \tag{21}$$

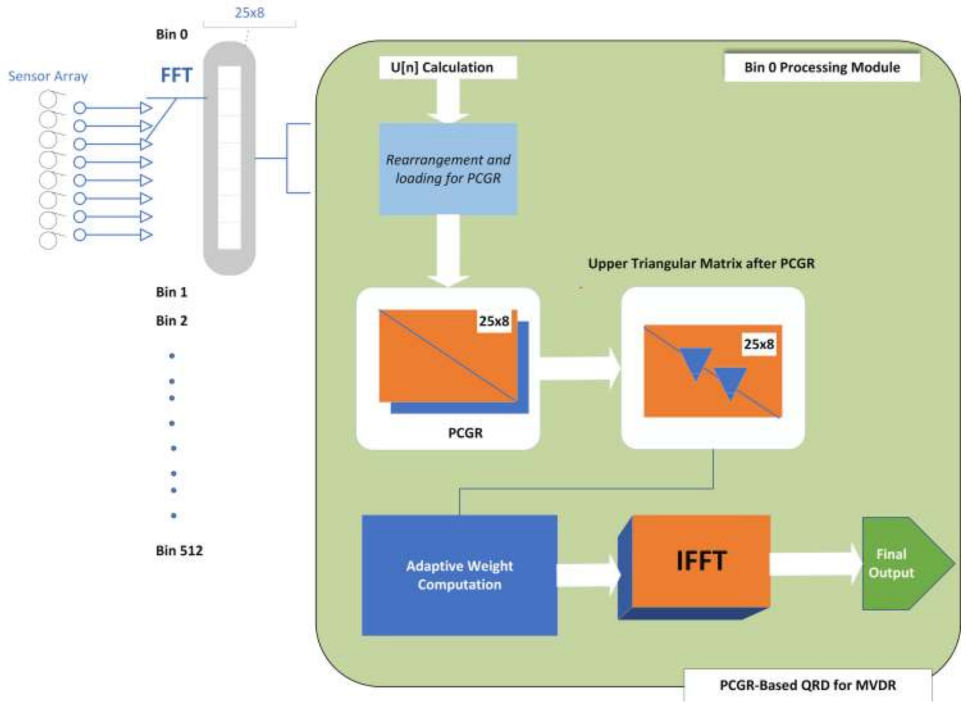


FIGURE 3. MVDR beamformer architecture.

By using all the above equations we get the value of T[n] get transformed sub array data vector and transformed steering vector respectively as

$$z_l[n] = S^{-T}[n]x_l[n] \tag{22}$$

$$a_z[n] = S^{-T}[n]a[n] \tag{23}$$

where S^T is a lower triangular matrix. We can simply compute $a_z[n]$

$$S^T[n]a_z[n] = a[n] \tag{24}$$

The beamformer output will now look below when we apply the forward substitution technique mentioned in the paper [12]. The QR Based MV Beamformer Block Diagram is explained in Figure 3. The PCGR Algorithm is explained in the later sections.

$$y[n] = w_z^H \frac{1}{M-L+1} \sum_{l=0}^{M-L} S^{-T}[n]x_l[n] \tag{25}$$

$$y[n] = w_z^H S^{-T}[n]x_{mean}[n] \tag{26}$$

The accelerated QR-based MVDR block diagram is given in Figure 4. We had used a Fast Fourier Transform (FFT) for converting time-domain data to the frequency domain and IFFT for back conversion. The algorithm for finding the output y[n] is described in figure 5.

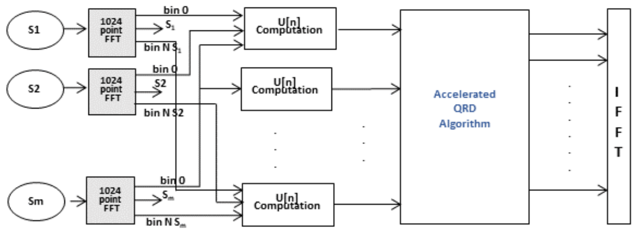


FIGURE 4. Accelerated QR-based MVDR block diagram.

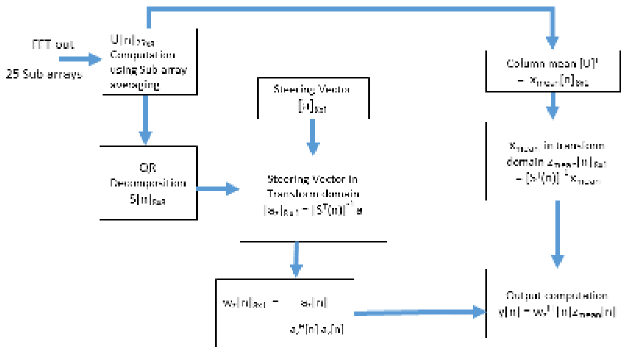

FFT out
25 Sub arrays
U[n]25x8 Computation using Sub array averaging
QR Decomposition S[n]8x8
Steering Vector [n]8x1
Steering Vector In Transform domain [az]8x1 = [S^T(n)]^-1 a
Column mean [U]^T = x_mean[n]8x1
x_mean in transform domain z_mean[n]8x1 = [S^T(n)]^-1 x_mean
wz[n]8x1 = az[n] / az^H[n] az[n]
Output computation y[n] = wz^H[n] z_mean[n]

FIGURE 5. Accelerated QR-based MVDR block diagram.

A. QR DECOMPOSITION ALGORITHMS

QR factorization/QU factorization is the commonly used terminology for QR Decomposition in Linear Algebra. This is the process by which a matrix A is decomposed into a product, which culminates to A = QR of an orthogonal matrix Q and an upper triangular matrix R. The QR algorithm is a very powerful algorithm to stably compute the eigenvalues and the corresponding eigenvectors/Schur vectors. For computing the QR Decomposition, we can use different algorithms like Householder transformations/Givens rotations [13], Gram–Schmidt process [14].

The method by which we calculate the U[n] is explained in the Figure 6.

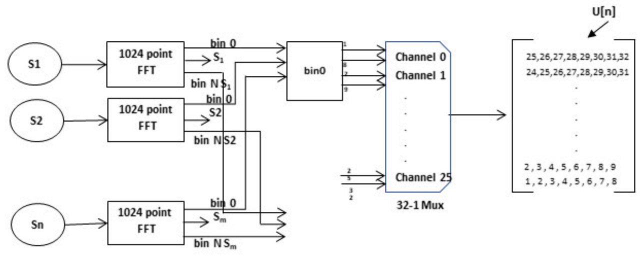


FIGURE 6. **Hardware realisation of *U*[*n*].**

1) SCHMIDT ORTHONORMALIZATION ALGORITHM

We have a set of input vectors (linearly independent) $v1, v\ 2, v\ 3\ldots, v\ n$. The algorithm [15] normalizes the first input vector

a: $u1 = normalize(v1)$

For the second u2vector to be orthogonal to the first one, we need to delete the v2parallel to u1part, which is a simple projection

b: $u_2 = normalize(v_2 - (v_2.v_1)u_1)$

We now have two orthonormal vectors; we now need to delete the parallel components of each of them and so on, for the third vector.

When applying this procedure on a system, the uk vectors are still not precisely orthogonal due to rounding errors. The lack of orthogonality is particularly bad for the "classical Gram – Schmidt" and we can claim that the classical Gram – Schmidt method is numerically unstable [16].

A minor change in the algorithm, which is referred to as MGS / modified Gram-Schmidt, will stabilize the Gram – Schmidt method [17]. The algorithm calculates uk according to the following equations.

$$u_k^{(1)} = v_k - Proj_{u1}(v_k) \tag{27}$$

$$u_k^{(2)} = u_k^{(1)} - Proj_{u2}(u_k^{(1)}) \tag{28}$$

$$\vdots$$

$$u_k^{(k-1)} = u_k^{(k-2)} - Proj_{u_{k-1}}(u_k^{(k-2)}) \tag{29}$$

The numerical complexity of the latter algorithm is floating-point operations asymptotically of $O(nk^2)$, where n is the vector dimensionality.

2) GIVENS ROTATIONS

Wallace Givens was the person who introduced this method to the world. Afterward, the method was named after him. The decomposition results are stored in arrays which originally store A, which will avoid us in using additional arrays in the givens rotation scheme.

Diverse uses for QR decomposition of Matrices are possible. It can be used to determine a matrix's eigenvalues to solve the so-called QR algorithm. Two rows of the matrix under transformation rotate on every step of the Givens Rotation scheme [18]. This transition parameter is chosen such that one of the entries in the existing matrix is eliminated. Initially, the first column entries are removed one by one, so the same is performed with the second column, etc., till column n − 1. The matrix which results is R.

There are two main stages in the algorithm: selecting the rotation parameter, and the second is rotation itself, which is done over two rows of the current matrix. The entries of these rows situated to the left of the pivot column are zero; thus, there is no need for modifications. The inputs in the pivot column are rotated at the same moment as the rotation parameter is selected. The second part of the move, therefore, consists of rotating two-dimensional vectors generated by the rotated row entries positioned on the right side of the pivot column [19]. So far as operations are concerned, modifying a column is equal to multiplying two complex numbers, one of these complex numbers is modulus 1. Two complex numbers will result in one subtraction, one addition, and four multiplication.

$$G(i,j,\theta) = \begin{bmatrix} 1 & 0 & 0 & .. & .. & .. & .. & 0 \\ 0 & 1 & 0 & .. & .. & .. & .. & 0 \\ . & . & . & .. & .. & .. & .. & . \\ . & . & . & c & .. & s & . & . \\ . & . & . & .. & .. & .. & .. & . \\ . & . & . & -s & .. & c & .. & . \\ . & . & . & .. & .. & .. & .. & . \\ 0 & 0 & 0 & .. & .. & .. & .. & 1 \end{bmatrix} \tag{30}$$

where $c = cos\theta$, $s = sin\theta$ and as we know $c^2 + s^2 = 1$

The product $G(i, j, \theta)X$ represents a counter-clockwise rotation of the vector X in the (i,j) plane of θ radians. QR decompositions can be computed with a series of Givens Rotations. Each rotation zeros an element in the sub diagonal of the matrix and forms the R matrix. The concatenation of all the Given Rotations forms the orthogonal Q matrix.

We will see an example of Givens Rotation. R calculation is explained as a flow [20]

we take the 3 × 3 matrix as follows

$$A = \begin{bmatrix} r_1 & r_2 & r_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \tag{31}$$

Givens Rotation Algorithm will generate the upper triangular matrix R from the above matrix by eliminating a_1, b_1 *and* b_2 in a three-stage of elimination.

Stage 1: The first Givens rotation matrix denoted by G1 and is defined as [21]

$$G1 = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{32}$$

where

$$c = \frac{1}{\sqrt{r_1^2 + a_1^2}} \tag{33}$$

$$s = \frac{a_1}{\sqrt{r_1^2 + a_1^2}} \tag{34}$$

The matrix G1 is multiplied with input matrix A to eliminate a_1 term. So the resulting matrix is G1*A

Stage 2: The second givens rotation matrix is denoted by G2 and is defined as

$$G2 = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ s & 0 & c \end{bmatrix} \tag{35}$$

where

$$c = \frac{\overline{r_1}}{\sqrt{\overline{r_1}^2 + a_1^2}} \tag{36}$$

$$s = \frac{b_1}{\sqrt{\overline{r_1}^2 + a_1^2}} \tag{37}$$

in the step 2 we multiply the matrix G2 with matrix obtained in stage 1 to eliminate b_1. So the resulting matrix is G2*G1*A

Stage 3: Givens rotation matrix G3 is defined as

$$G3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{bmatrix} \tag{38}$$

where

$$c = \frac{\overline{a_2}}{\sqrt{\overline{a_2}^2 + b_2^2}} \tag{39}$$

$$s = \frac{\overline{b_2}}{\sqrt{\overline{a_1}^2 + b_2^2}} \tag{40}$$

here in step 3 we multiply the matrix G3 with matrix obtained in stage 2 to eliminate b_2. Resulting matrix is the upper triangular matrix, our R

$$R = G3^*G2^*G1^*A \tag{41}$$

3) MODIFIED SQUARED GIVENS ROTATIONS

QR decomposition of a matrix A results in two matrices Q and R A=QR Where Q is an orthogonal matrix and R is an upper triangular matrix. The matrix A can be also be decomposed using Squared Givens Rotation (SGR) [21]as

$$A = Q_A D_U^{-1} U \tag{42}$$

where $Q_A = QD_R$

$$D_U = D_R^2 \tag{43}$$

$$U = D_R R \tag{44}$$

$$D_R = diag(R) \tag{45}$$

$$D_U = diag(U) \tag{46}$$

MSGR Algorithm will be explained by taking a 3 × 3 matrix of complex values as below

$$\begin{bmatrix} r \\ a \\ b \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \tag{47}$$

MSGR generates the upper triangular matrix U from the above matrix by eliminating a_1, b_1 *and* b_2 in a three stage process.

Stage 1:

Rotate rows r and a to eliminate element a_1.

Introducing u, which is defined [21] by $u = r_1^* r$

$$\overline{u} = u + a_1^* a$$

Similarly introducing v as

$$v = W_a^{(-\frac{1}{2})} a \tag{48}$$

where $w_a > 0$ is a scalar factor.

$$\overline{v} = v - \frac{v_1}{u_2} u \tag{49}$$

Then back conversion from V space is defined as

$$\overline{a} = W_a^{\frac{1}{2}} \overline{v} \tag{50}$$

where

$$\overline{W_a} = W_a \frac{u_1}{\overline{u_2}}$$

Stage 2:

Rotate r and b to eliminate $b_1.\overline{r}$ is already in U space. The row b is translated to V space by

$$v = \overline{W_a}^{\frac{1}{2}} b \tag{51}$$

v_b is updated as

$$\overline{v_b} = v_b - \frac{v_{b1}}{\overline{u_1}} \overline{u} \tag{52}$$

Then back conversion from V space from equation 42 is as follows

$$\overline{b} = \overline{W_b}^{\frac{1}{2}} \overline{v_b} \tag{53}$$

where $\overline{W_b} = W_b \frac{\overline{u_1}}{u_1}$

Stage 3:

Rotate $\overline{a}$ *and* $\overline{b}$ in order to eliminate $\overline{b_2}$. $\overline{a}$ must be translated to U-space and $\overline{b}$ in to V-space.

$$v_a = \overline{a_2}^* \overline{a}$$

Then update u_a and v_b. Finally, the last row must be translated to U-space to get matrix U. The MSGR method for a general case is sited in the paper [21] as below

$$\bar{u} = u + wv_k^* v \bar{v} = v - (\frac{v_k}{u_k})^* u \bar{w} = w \frac{u_k}{\bar{u_k}} \tag{54}$$

Finally the matrix R is obtained from matrix U.

4) FAST TEOPLITZ ORTHOGONALIZATION

A matrix T is Toeplitz if the elements are constant on-diagonal. Because a Toeplitz matrix is calculated by a limit of 2n-1 numbers, here are algorithms that solve Toeplitz linear equation structures using just $O(n^2)$ operations, or even $O(n\ log^2\ n)$ while implementing quick techniques [21].

Consider Martix A

$$A = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & .. & .. & t_{-n} \\ t_1 & t_0 & t_{-1} & .. & .. & t_{1-n} \\ t_2 & t_1 & t_0 & .. & .. & t_{2-n} \\ t_m & t_{m-1} & t_{m-2} & .. & .. & t_{m-n} \end{bmatrix} \tag{55}$$

The size of Toeplitz matrix A is (m+1)x(n+1). Shift invariance property of the Toeplitz matrix can be utilized to solve our problem of QR Decomposition. The paper [21] describes how the Matrix A can be partitioned, and it suggests two methods to do it.

Matrix A's QR Decomposition is given by A=QR

$(R_{-1})^T a = (A_{-1})^T x$, Where R is upper triangular matrix of order n+1. The calculation of R consists of two stages [35].

$$A_{-1} = P_{-1} \begin{bmatrix} R_{-1} \\ 0 \end{bmatrix} \tag{56}$$

The above equation describes QR decomposition of mxn matrix, where P is an orthogonal matrix of order m and R is an upper triangular matrix of order n [35]. The actual method is shown below in figure 7

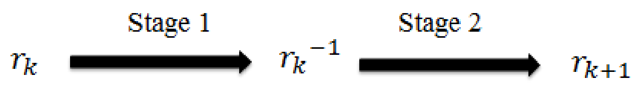


FIGURE 7. **Fast teoplitz orthogonalization.**

Stage 1 will compute the r_k^{-1} and w_k from r_k and w_1,w_2,....w_{k-1}.

Stage 2 will compute r_{k+1}, U_k and V_k from first determined the k^{th} column of R and V_k using equations.

IV. ACCELERATED QRD-MV BEAMFORMER IMPLEMENTATION USING CGR ALGORITHM

CGR-QRD has less number of multiplications than the GR implementations in the literature [22]. Hence, it requires fewer resources to perform the decomposition. Apart from reduced computational complexity, CGR-QRD can exploit more fine-grained and coarse-grained parallelisms compared to other native approaches. The 2D systolic array architecture for Column wise givens rotation depicts the involvement of much overlapping in computation and communications between the processing elements, which is described in figure 8.

Here in this section we describe the QRD-MV Beamformer implementation [12] using CGR Algorithm described in the paper [26]. The process is explained in below

Algorithm 1 Accelerated QRD-MV Beamformer

```
Input: U[n], a[n]
Input: S[n] = QR Decomposition of U[n] method used is
    CGR
Input: $a_z[n]$= forward substitute ($s^T[n]$, $a[n]$)
Input: $w_z[n] = \frac{a_z[n]}{a_z H[n] a_z[n]}$
    $x_{mean}[n] = Column\ mean(U^T[n])$
    $z_{mean}[n] = forward\ substitute(S^T[n], x_{mean}[n])$
    We need to calculate the upper triangular matrix using
    the Column wise given rotation algorithm.
    We need to calculate vector x which is $Lx = b$ by using
    the forward substitution technique, where L is lower
    triangular martix.
    We need to calculate the mean of the columns of the
    matrix U[n].
Output: $y[n] = w_z^H[n] z_{mean}[n]$
```

Acceleration process is explained in the literature survery now we will see how we use the Column wise givens rotation to find the upper triangular matrix and how to is more efficent that the standard methods.

Consider the non-singular matrix X of size n x n which is given as

$$R = Q^T X \tag{57}$$

where $QQ^T = I$ and $Q = Q_{n-1} \ldots\ldots Q_2 Q1$, I is an Identity Matirx and $Q_k = G^k_{k,k+1} \ldots G^k_{n-2,n-1}$, $1 <= k <= n-1$ where $G_{i,j} = diag(I_{i-2}, \bar{G}_{i,j}, I_{m-1})$ and

$$\bar{G} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

$c = \frac{A_{i-1,j}}{t}$, $s\frac{A_{i,j}}{t}$ and $t = \sqrt{A^2_{i-1,j} + A^2_{i,j}}$ if we denote GR_M as the number of multiplication operators required for Givens Rotation and CGR_M as that of Column Wise Givens Rotations, we will culminate to the below findings.

$$CGR_M = \frac{2n^3 + 3n^2 - 5n}{2} \tag{58}$$

$$GR_M = \frac{4n^3 - 4n}{3} \tag{59}$$

we take the ratio of 46 and 47,

$$\beta = \frac{CGR_M}{GR_M} = \frac{3(2n+5)}{8(n+1)} \tag{60}$$

If n goes to infinity, β will become $\frac{3}{4}$, ie, if we increase the size of the matrix, the number of multiplications in CGR will become $\frac{3}{4}$ times that of Givens Rotations, which will reduce

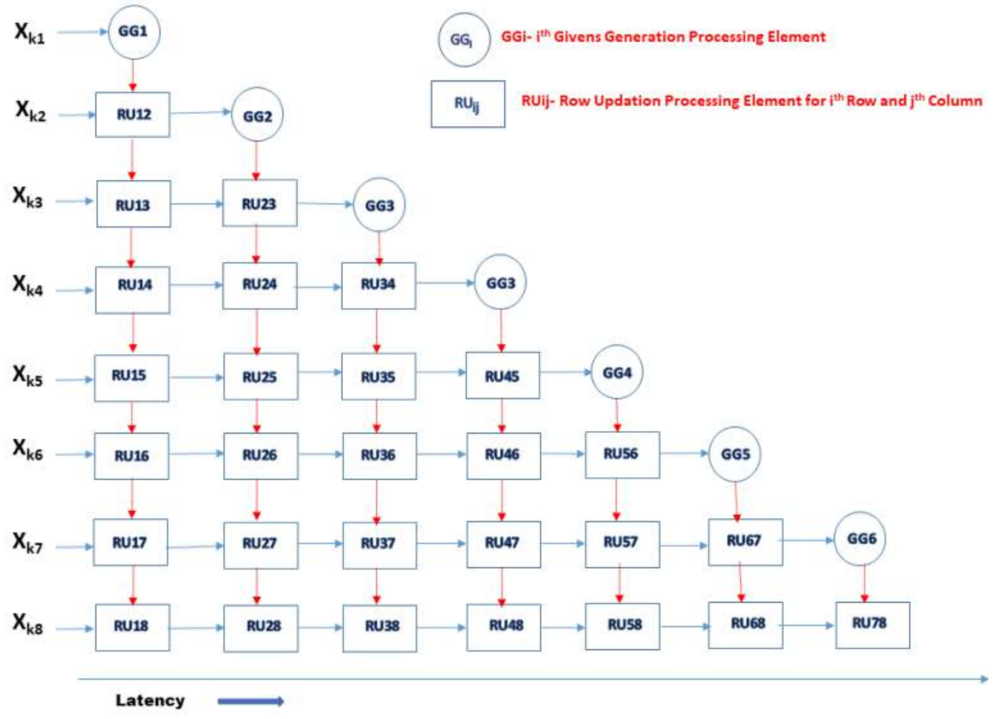


FIGURE 8. Scheduling of operations for CGR.

the complexity of Operations [22]. Many of the previous studies have shown successful results in reducing the computational cost, but all of them are often viewed as approximate implementations of the traditional MV beamformer. Here we propose a computationally superior QRD based MV beamformer, which is mathematically like the traditional MV beamformer [12]. The algorithm of the CGR based MVDR beamformer is explained below.

V. THE PROPOSED ARCHITECTURE- PIPELINED COLUMN WISE GIVENS ROTATION (PCGR)

We had explained about the QRD based MV beamformer in the earlier sections. The spatial covariance matrix and its Inverse is not required for us here, but significant calculations are still necessary to perform calculations for QR Decomposition. Results from paper [12] have already described the method of finding the beamformed output by using U[n] and S[n] without having to calculate Q[n]. Acceleration method also is depicted in paper [12] which is adapted in our study also, and the computations are made superior in our study were we propose a new architecture for finding the R. The scheduling of CGR operations for FPGA implementation in the proposed system is shown in Figure 11 as new scheme named as PCGR (Pipelined Column Wise Givens Rotation).

Here in this section, we propose a new architecture for finding the Column Wise Givens Rotation in a superior method. The architecture of the Givens Generation [23], [24] is modified with memory elements to store all the intermediate values which have enabled us to pipeline the entire architecture, thereby saving the clock cycles to complete an

Algorithm 2 CGR MVDR Beamformering Algorithm

```
1: Initializations: FFT defining N=1024 Sensors used M=32; Subarray Length used L=8;Overlap O=128; Number of scan lines =61;
2: Operations 3 to 6 to be performed for each scan line: • We need to perform 1 K Point FFT with N-O Channels and O old samples, for M Channels. •For a focal point perform delay compensation for each subband by multiplying with steering vector calculated for the point. •Grouping of subbands are required, Subband0(bin0) to subbandN/2(binN/2) of 0 to M-1 Channels is explained Bin0=[$F_0(0), F_1(0), F_2(0) \ldots\ldots F_{M-1}(0)$]
   Bin1=[$F_0(1), F_1(1), F_2(1) \ldots\ldots F_{M-1}(1)$]
   BinN/2=[$F_0(N), F_1(N), F_2(N) \ldots\ldots F_{M-1}(N)$]
   •Across each subband group, perform the subarray averaging and generate the R Matrix using CGR Algorithm •Compute adaptive weights and output • Align 1024 frequency bins and feed to IFFT and generate time domain data for each IFFT frame replace previous 128 samples with new samples.
3: Generate the scan line by repeating the above operations.
4: Align all 61 scan lines and perform dynamic range compression, bilinear interpolaration and generate an Ultrasound Frame.
5: Perform steps 1 to 4 to generate a minimum of 30 frames in one second to get a good ultrasound image.
```

iteration, the value paid for this is extra flops which are used to store the intermediate values. The modified Given generation

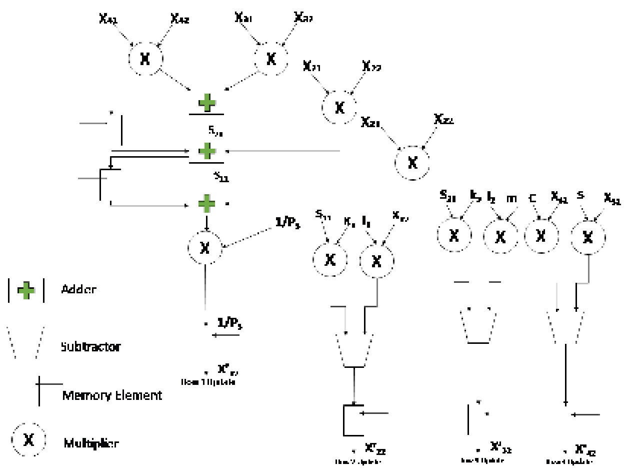


FIGURE 9. Proposed architecture of the row updation processing element-RU.

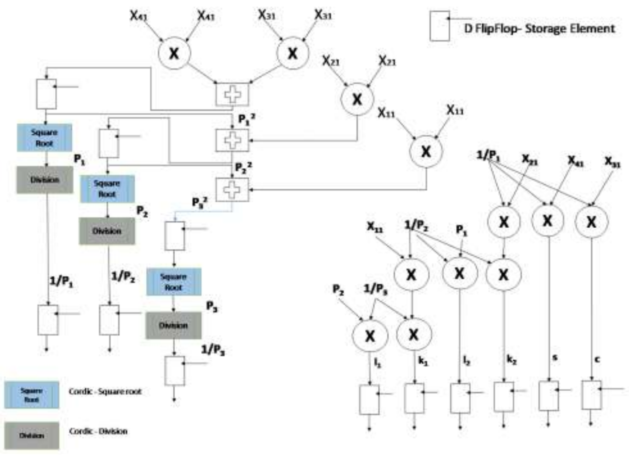


FIGURE 10. Proposed architecture of the given generation processing element-GG.

architecture is mentioned below in Figure 10, and the Row updation modified architecture is shown in Figure 9. The Row updation architecture [25], [26] is modified with pipelining features that enable us to save the time of computations. The new architecture is shown in Figure 11. We use a system clock of 20 MHz, which is multiplied using the FPGA Clocking resources to achieve a 200 Mhz clock, which is used to run the modified algorithm's adaptive weight calculation.

A. ARCHITECTURE AND THE WORKING

The Architecture of column-wise givens rotation is adopted from paper [21], which is modified and implemented on FPGA utilizing the parallel processing capabilities of FPGA. The proposed architecture, whose work is explained below for a 4 × 4 Matrix, can be extended to any size of the matrix. Column 1 is given as input to GG1. All the operations in GG1, as shown in the architecture below, require only column 1. The outputs obtained have to be stored in memory, as shown in figure 11. The first column of the matrix becomes zero after GG1 except for the first element, p3 (obtained from GG1), as shown above.

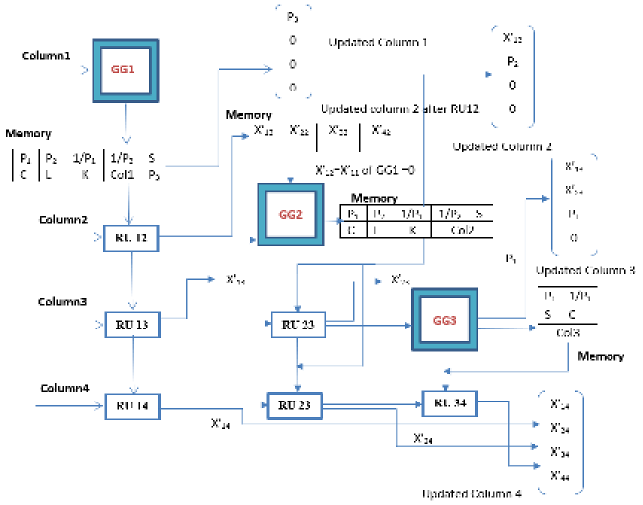


FIGURE 11. Operation architecture for a 4 × 4 matrix.

Parallelly RU12 can begin the row 1 update for column 2, followed by row4 update, row3 update, and row2 update for column 2. The inputs to RU12 is column 2 and column 1, along with other outputs, which is already stored in memory. The output obtained is updated in column 2. The first element of this updated column 2, i.e., X'_{12} has to be stored separately since it remains unchanged in the final output matrix—the rest of the elements changes in GG2.

The input to GG2 is this updated column 2. This takes the place of column 1 as it was given as input to GG1 earlier. Here, the first element, i.e., X_{12}, has to be made zero so that the same architecture can be used. As compared to GG1, the difference is X_{41} becomes X_{42}, X_{31} becomes X_{32}, X_{21} becomes X_{22} and X_{11} becomes X_{12} which is made zero already. The outputs of GG2 operations and the updated column2, also have to be stored in memory, as shown above. The second column of the final output matrix is ready here, the first element of which was already obtained from RU12 and stored earlier. The second element will be p2 (obtained from GG2) and the rest zeros, as shown in figure 11.

The inputs to RU13 is column 3 and column 1, along with other outputs, from memory. Just like RU12, RU13, too, can begin the row 1 update for column 3, parallelly with GG1 operations, followed by row4 update, row3 update, and row2 update for column 3. The output obtained is updated in column 3. The first element of this updated column 3, i.e., X'_{13}, has to be stored separately since it remains unchanged in the final output matrix—the rest of the elements changes in the next update in RU23 and GG3.

The inputs to RU23 is updated column 2 from GG2 in which X12 was already made zero and updated column 3 from RU13, along with other outputs from the memory of GG2. The first element of this updated column 3 can be made zero, but not compulsory, since X_{12} is already zero. RU23 can begin the row 2 update for column 3, parallelly with GG2 operations, followed by row4 update and row3 update for column 3. The output obtained is updated in column 3. The output of the row 2 update, i.e., X'_{23} has to be stored separately since it remains unchanged in the final output matrix—the rest of the elements changes in GG3. The input to GG3 is this updated column 3, with X_{13} and X_{23} made zero. The same architecture can be used as explained above for GG2. The outputs of GG3 operations, as well as the updated column3, also have to be stored in memory, as shown above. The third column of the final output matrix is ready here, the first and second elements of which were already obtained from RU13 and RU23 and stored earlier. The third element will be p1 (obtained from GG3) and the last element zero, as shown in figure 11.

The inputs to RU14 is column 4 and column 1, along with other outputs, from the memory of GG1. Just like RU12 and RU13, RU14, too, can begin the row 1 update for column 4, parallelly with GG1 operations, followed by row4 update and row3 update for column 4. The output obtained is updated in column 4. The first element of this updated column 4, i.e., X'_{14} has to be stored separately since it remains unchanged in the final output matrix. The rest of the elements changes in the next update in RU24 and RU34.

The inputs to RU24 is updated column 2 from GG2's memory, in which X12 was already made zero and updated column 4 from RU14, along with other outputs from the memory of GG2. The first element of this updated column 4, can be made zero, but not compulsory since X12 is already zero. RU24 can begin the row 2 update for column 4, parallelly with GG2 operations, followed by row4 update and row3 update for column 4. The output obtained is updated in column 4. The output of the row 2 update, i.e., X'_{24} has to be stored separately since it remains unchanged in the final output matrix—the rest of the elements changes in RU34. The inputs to RU34 is updated column 3 from GG3, in which X_{13} and X_{23} was already made zero, and updated column 4 from RU24, along with other outputs from the memory of GG3. RU34 can begin the row 3 update for column 4, parallelly with GG3 operations, followed by row4 update for column 4. The output obtained is updated in column 4. The output of the row 3 update is X'_{34}, and the output of row 4 update is X'44. The resultant matrix is shown below. The architecture of a 4×4 Matrix is shown in Figure 11. The architecture was finding the upper triangular matrix using PCGR shown in Figure 14. The algorithm of the new architecture is explained below. The working of the FSM is stated in the algorithm, which enables us to save the various repeated multiplications and divisions as we reuse the GG and RU blocks, the additional hardware required here are the memory and FIFO blocks.

The architecture is summerized in figure 12.

VI. EXPERIMENT SETUP

The complete experiment setup consists of a Custom Transceiver 128-Channel Board shown in Figure 15, High-End FPGA Prototyping Board (DBF Board) and a PC. Both boards are connected using Board to Board connector.

Algorithm 3 Pipelined Column Wise Givens Rotation - PCGR

```
Input: Matrix should be saved to memory intially.
Input: FSM should generate inputs for fetching required
    values to FIFO from memory which is given as input to
    GG and RU Blocks.
Input: FSM should generate appropriate r/w signal to write
    outputs from GG and RU Blocks to memory.
    STATE A-Reset STATE
    STATE B-Memory reading and writing to FIFO Control
    Signals generation
    STATE C-Reading from FIFO and GG Execution i=1 to
    3 GGi
    STATE D-Writing output from GGi to Memory
    STATE E-Writing values from Memory to FIFO and
    input to RU Block.
    STATE F-Execution of RU Block and Writing out to
    Memory.
    STATE B- if the $i <= 3$ or else STATE G
    STATE G - Output STATE. Output Matrix is written into
    Memory.
Output: Upper Triangular Matrix
```

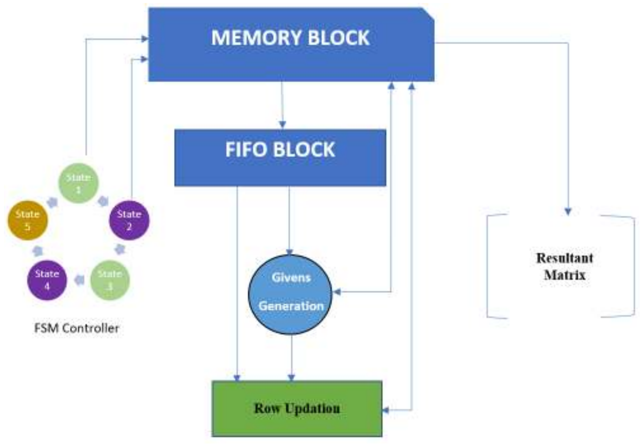


FIGURE 12. Architecture for finding the upper triangular matrix using PCGR.

The ultrasound transducer probe is connected to the Transceiver board using the probe connector. The 128 channel Ultrasound Transceiver board has high voltage transmit pulsars to energize up to 128 transducers and 128 channels analog frontend (AFE) signal conditioning circuits. The board moreover contains a Xilinx Kintex-7 XC160T FPGA, which acts as the transmit-receive controller and transmit beamformer for the system. The board also has an appropriate clock and control tree structures for clock and power administration. The high voltage transmit pulsars energizes the transducer cluster through FPGA controlled (transmit beamformer), and the received echoes are analog signal conditioned and gushed to a 128 channel Digital Beamformer (DBF) board to perform receive beamforming.

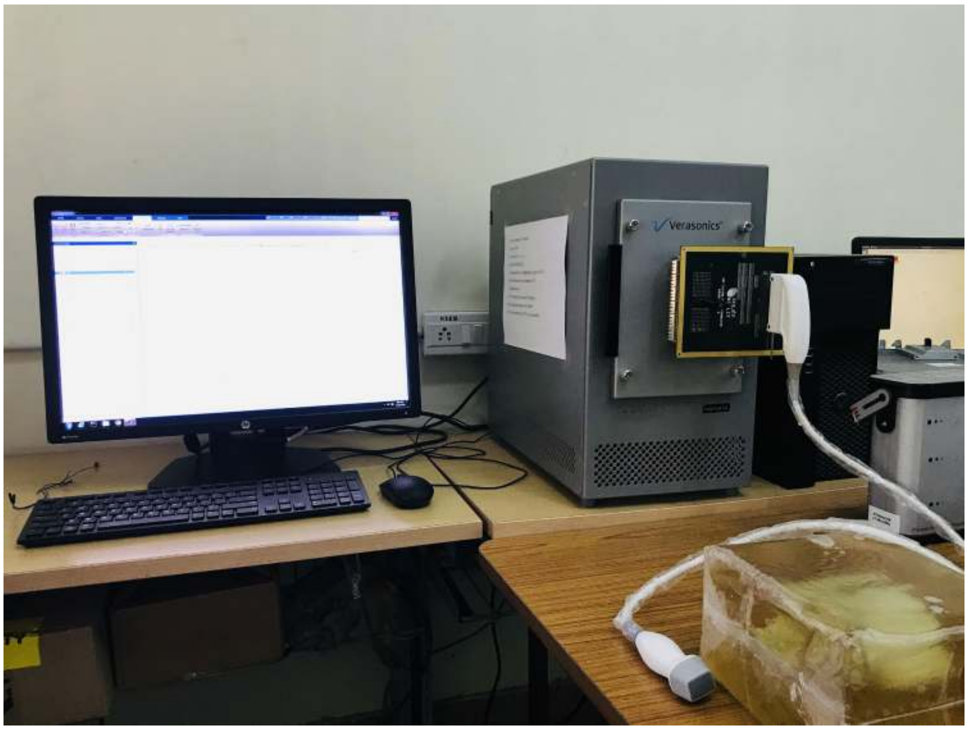


FIGURE 13. Ultrasound research platform.

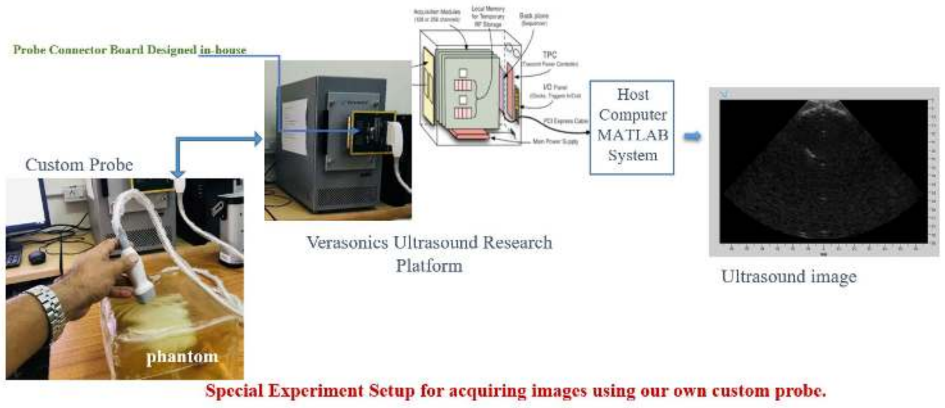

Probe Connector Board Designed in-house
Custom Probe
phantom
Acquisition Modules (128 or 256 channels)
Local Memory for Temporary RF Storage
Back plane (Sequencer)
TPC (Transmit Power Controller)
I/O Panel (Clocks, Triggers In/Out)
PCI Express Cable
Main Power Supply
Host Computer MATLAB System
Verasonics Ultrasound Research Platform
Ultrasound image
Special Experiment Setup for acquiring images using our own custom probe.

FIGURE 14. The whole setup for the experiment.

The DBF board is engineered based on Xilinx-Kintex-7 410T FPGA, which has sufficient assets for versatile array signal processing algorithms for QRD-MV Beamformer execution. The Transceiver board has a high voltage transmitter section which transmits the ultrasound pulses. The pulses goes to the probe connected to the board, which produces echoes after hitting targets from the cyst phantoms. The echoes are captured and processed by the transceiver analog front end section and transferred to the Digital Beamformer Board(DBF). The DBF Board on which we had implemented the QR-MV Beamformer algorithm. The beamformed data is transferred to the PC via Gigabit Ethernet and processed by the custom made software and the ultrasound image is formed.

The data for second experiment was collected from verasonics^{TM} ultrasound research platform [27] shown in figure 13 and using the phased array 64 element probe and the data is processed in MATLAB.

The Setup for validating the results consists of a Verasonics^{TM} Ultrasound Research Platform, which is unique hardware that takes real data from the phantom through their ultrasound probe and process the data using Beamforming Algorithms which are in build and form the images using their image forming tools. This Research Platform has a PC

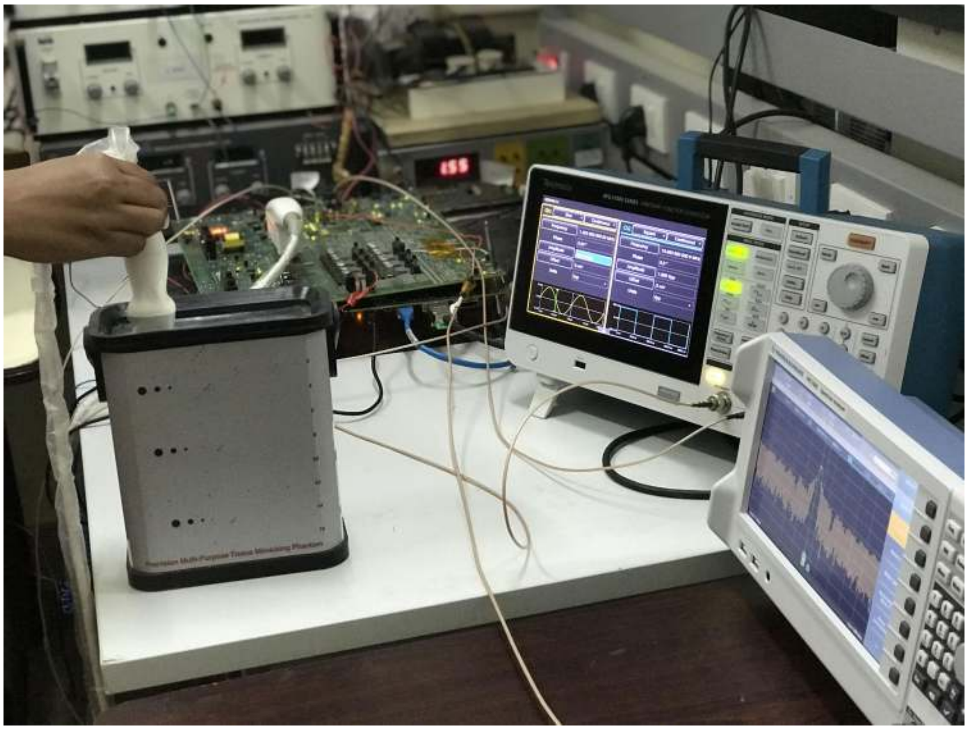


FIGURE 15. The setup of the implementation.

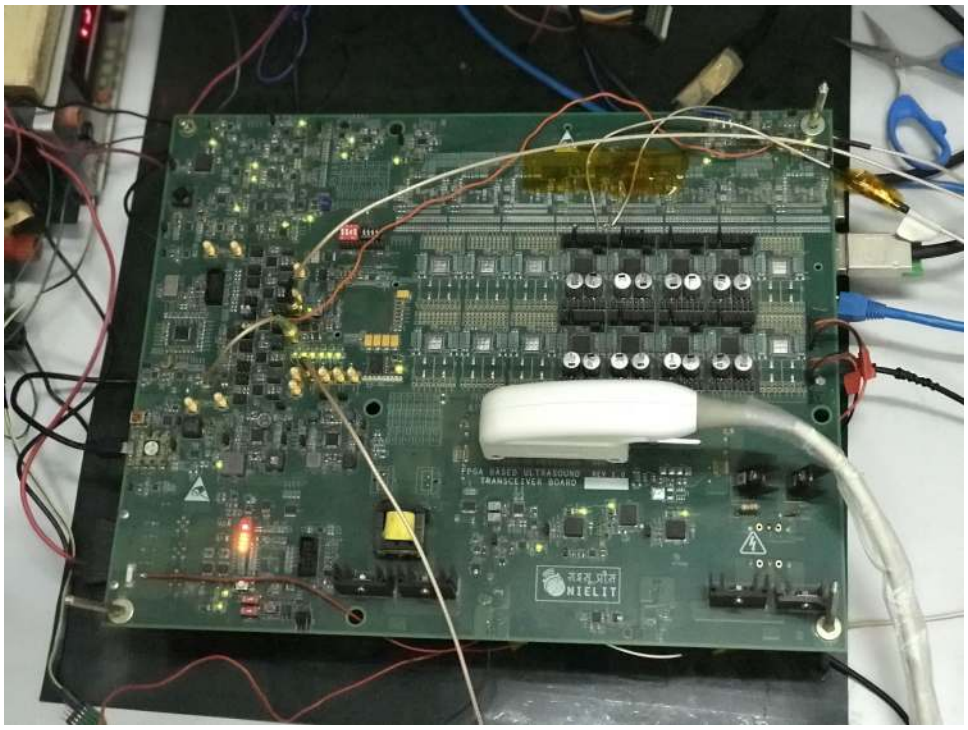


FIGURE 16. The custom ultrasound transceiver board.

connected to it, and this unit is a flexible tool for transmitting, receiving, and processing ultrasound information. Amplification, sampling, and filtering operations are performing on receiving ultrasound data. Finally, the vantage unit stores the data and handovers to host the computer via PCI express cable [27]. Another important piece of equipment that helps

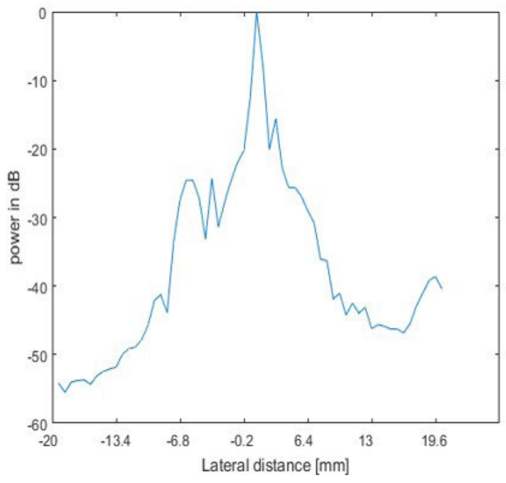


FIGURE 17. Beam pattern of DAS beamformer.

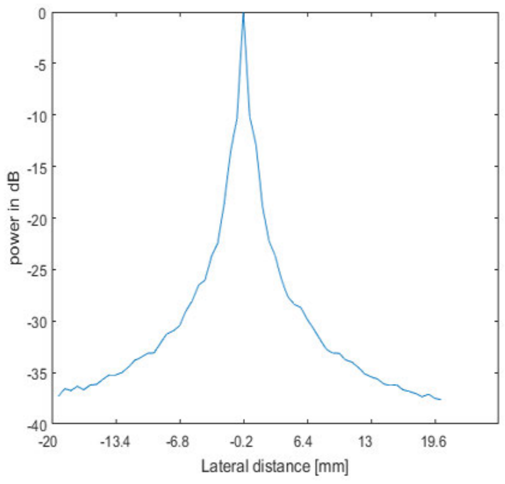


FIGURE 18. Beam pattern of PCGR-MVDR beamformer.

TABLE 1. Computational complexity of accelerated MVDR beamformer.

Steps of Calculation	Accelerated MVDR
S[n]	Depends on QR decomposition Methods*
$a_z[n]$	$\frac{L^2+2}{2}$
$w_z[n]$	2L
$x_{mean}[n]$	L
$z_{mean}[n]$	$\frac{L^2+2}{2}$
y[n]	L

TABLE 2. PCGR performance summary.

PRF	4KHz
Sampling Frequency	20MHz
No. of beams per frame	61
Time to process one beam	0.25msec
Time to process one frame	14.6msec
Frames per Second as per FPGA performance	68.3 fps

in collecting real data is the target Phantoms. There are different phantoms like Kidney Phantom, Resolution Phantom, etc.. Resolution Phantom is a rectangular box of dummy bodies containing numerous cysts. The cyst's locations are known and clearly defined. This information is also provided to the user on the phantom, and this allows the user to fine-tune/validate his/her algorithm with the resolutions obtained from the real data.

The real experiment setup is shown in Figure 15.

VII. RESULTS AND DISCUSSIONS

The proposed PCGR Architecture is implemented on Kintex 7 custom board using Verilog HDL. The RTL Simulation, Synthesis and Implementation was carried out on Custom Designed FPGA Boards and results were validated with results from Ultrasound Research Platform.

A. FPGA IMPLEMENTATION

The implementation of the accelerated QRD-MV beamformer was completed on the Prototype setup mentioned earlier in the experimental setup section. We had completed simulations and implementations to validate Algorithm's behavior and performance [30]. We had implemented the Algorithm on Custom FPGA Board, which is called as Digital Beamformer Board (DBF), which is connected to PC via Gigabit Ethernet port, and the data from sensor come from to this DBF Board through the board to board connector. The resource utilization for XC7K410T FPGA is detailed in Table 4 and Table 3. The resource utilization was on the higher side compared with standard implementations as in our work we had used pipelining to improve the latency and clock frequency the price paid for this is the extra hardware incurred to implement the pipeline. We used Xilinx LogiCORE IP to implement the Fast Fourier Transform for the architecture [36] as shown in Figure 3.

B. VALIDATION OF RESULTS

We have used the Verasonics Research Platform to validate our results. The Vantage Research Ultrasound Platform uses proprietary hardware and software technologies to provide direct access to raw ultrasound data, while preserving the ability to perform high-quality real-time imaging with custom software, at clinically useful frame rates. We had used P4-2v phased array 64-element phased array probe for our research. It has a pitch of 0.3mm, Elevation focus of 50-70mm, sensitivity of -64-95 dB. We used Precision Multi-purpose Tissue mimicking Resolution Phantom for our experiment to identify the algorithm's performance for resolution. The proposed architecture of MVDR was validated on real data captured from resolution phantom using Ultrasound Research Platform Vantage-64. As Figure 20 explains that the

TABLE 3. Comparison of proposed work with previous works.

Scheme	Latency/ns	Device	Area /LUT	Algorithm	Accuracy	Frames per second(fps)
The proposed work	225	XC7K410T	34567	PCGR	Very Good	68.3
J.U.KIDAV[31]	201	Xilinx Kintex-7	26 485	DCD	Very good	65.5
WANG D et al.[32]	167	Altera Straitix	4 863	CORDIC	Very good	-
EDMAN F et al.[33]	80	Xilinx Virtex-II	117 (slices)	Smith	Poor	20
LIU J et al. [34]	620	Xilinx XC2VP30	527 (slices)	DCD	Very good	-
Junying Chen et al. [37]	580	Xilinx XC6VLX240T	21890	Toeplitz MVDR	Good	71.2
Shin-Shiang Wang et al. [30]	210	Xilinx Virtex XC7z045	23789	MVDR with Matrix Inversion	Good	20

TABLE 4. FPGA resource utilization for 32 channel QRD-MV beamformer.

Hardware Resources	Utilized	Utilization in percentage
No. of DSP48E1s	1283	83.31
No. of Block RAM/FIFO (36Kb)	402	50.53
No. of slice LUTs	184549	45.37
No. of slice Registers	232808	45.79

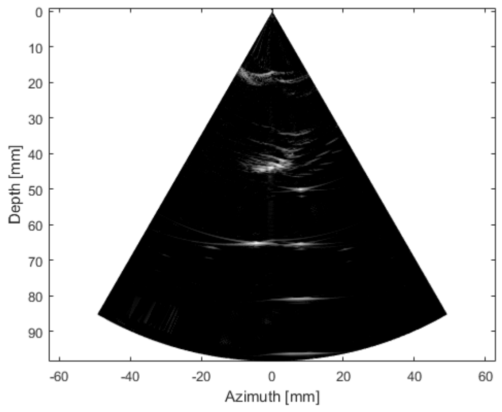


FIGURE 19. Multiple cyst image of DAS beamformer.

CGR -MVDR Beamformer has more resolution compared to DAS Beamformer in Figure 19.

Simulations and Implementations were performed to validate the idea put forward in this article. The performances of MV and QRD-MV beamformers are similar; the parallel pipelined architecture has improved the performance of beamformers in achieving higher frames per second(fps). The beam pattern of conventional beamformer-DAS and accelerated QRD-MVDR beamformer for a single cyst at a zero degree angle are shown in Figure 17 and Figure 18.

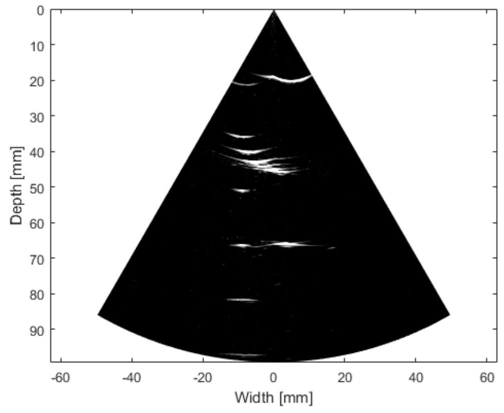


FIGURE 20. Multiple cyst image PCGR based MV beamformer.

The beamwidth becomes narrow in accelerated QRD-MVDR than conventional by a factor of 0.6 mm. Images from the Research Platform for DAS beamformer and accelerated QRD-MV beamformer for multiple cysts are shown in figure 19 and figure 20, respectively. The cysts are separated in figure 20 more than figure 19.

C. COMPARISON WITH LITERATURES

Parallelism and Pipelining techniques have been utilized to make the implementation faster and less complicated by operations. MVDR and QRD-MVDR beamformers are the same and one in terms of performances. FPGA Implementation of the PCGR based QRD on Kintex 7 FPGA is carried to compare the results with that from the Research Platform. The resource utilization of PCGR with previous works is tabulated in Table 3. The previous works related to MV-Beamformer has been compared in the table mentioned above, works [37]–[39] are taken for comparison here, which has comparable results with our algorithm. The computational complexity of accelerated MVDR beamformer based on QR decomposition is shown in Table 1.

VIII. CONCLUSION

In this work, an accelerated QRDMV beamformer using the column-wise given rotation algorithm is implemented on the custom transceiver board Figure 16. We have also implemented the PCGR based QR-MV Beamformer, which reduces the computational time, as we modified the architecture of the conventional CGR Algorithm [21] for porting it on the FPGA. The performance summary of the algorithm is pictured in Table 2. For hardware optimization, the computational blocks employed in the PCGR blocks were effectively reused by adding extra memory and control logic. The architecture is implemented on Verasonics Ultrasound Research Platform, in vitro experiments, was carried out and concluded that the image quality is superior to the standard methods. It is evident from the implementation that the number of clock cycles required to form a beam depends on the adaptive weight vector generation time, and it increases as the number of channels increases. Real-time performance is investigated on FPGA Prototype Lab Model and results are comparable with MATLAB MVDR Model ported on Verasonics Vantage-64 ultrasound research platform.

REFERENCES

[1] T.-J. Shan and T. Kailath, ''Adaptive beamforming for coherent signals and interference,'' *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 3, pp. 527–536, Jun. 1985.

[2] J.-F. Synnevag, A. Austeng, and S. Holm, ''Benefits of minimum-variance beamforming in medical ultrasound imaging,'' *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 56, no. 9, pp. 1868–1879, Sep. 2009.

[3] C. A. Balanis and P. I. Ioannides, *Introduction to Smart Antennas*, 1st ed. San Rafael, CA, USA: Morgan & Claypool, 2007.

[4] O. L. Frost, ''An algorithm for linearly constrained adaptive array processing,'' *Proc. IEEE*, vol. 60, no. 8, pp. 926–935, Aug. 1972.

[5] L. J. Griffiths and C. W. Jim, ''An alternative approach to linearly constrained adaptive beamforming,'' *IEEE Trans. Antennas Propag.*, vol. 30, no. 1, pp. 27–34, Jan. 1982.

[6] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.

[7] J. An and B. Champagne, ''GSC realisations using the two-dimensional transform-domain LMS algorithm,'' *IEE Proc.-Radar Sonar Navigat.*, vol. 141, no. 5, pp. 270–278, 1994.

[8] J. Capon, ''High-resolution frequency-wavenumber spectrum analysis,'' *Proc. IEEE*, vol. 57, no. 8, pp. 1408–1418, Aug. 1969.

[9] G. O. Glentis, ''Implementation of adaptive generalized sidelobe cancellers using efficient complex valued arithmetic,'' *Int. J. Appl. Math. Comput. Sci.*, vol. 13, no. 4, pp. 549–566, 2003.

[10] W. F. Gabriel, ''Adaptive arrays—An introduction,'' *Proc. IEEE*, vol. 64, no. 2, pp. 239–272, Feb. 1976.

[11] H. Singh and R. M. Jha, ''Trends in adaptive array processing,'' *Int. J. Antennas Propag.*, vol. 2012, pp. 1687–5869, Feb. 2012, doi: 10.1155/2012/361768.

[12] J. Park, S.-M. Wi, and J. S. Lee, ''Computationally efficient adaptive beamformer for ultrasound imaging based on QR decomposition,'' *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 63, no. 2, pp. 256–265, Feb. 2016.

[13] W. Givens, ''Computation of plain unitary rotations transforming a general matrix to triangular form,'' *J. Soc. Ind. Appl. Math.*, vol. 6, no. 1, pp. 26–50, Mar. 1958.

[14] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[15] Y. Chu and W.-H. Fang, ''A novel wavelet-based generalized sidelobe canceller,'' *IEEE Trans. Antennas Propag.*, vol. 47, no. 9, pp. 1485–1494, Sep. 1999.

[16] M. Karkooti, J. R. Cavallaro, and C. Dick, ''FPGA implementation of matrix inversion using QRD-RLS algorithm,'' in *Proc. Conf. Rec. 39th Asilomar Conf. Signals, Syst. Comput.*, 2005, pp. 1625–1629.

[17] C. Singh, S. Prasad, and P. Balsara, ''VLSI architecture for matrix inversion using modified gram-Schmidt based QR decomposition,'' in *Proc. 20th Int. Conf. VLSI Design Jointly 6th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2007, pp. 836–841.

[18] F. Edman and V. Owall, ''A scalable pipelined complex valued matrix inversion architecture,'' in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 4489–4492.

[19] C. Jian-Shu and W. Xue-Gang, ''LSMI algorithm based on inverse QR decomposition,'' in *Proc. Int. Conf. Commun., Circuits Syst.*, Jun. 2006, pp. 262–265.

[20] L. Ma, K. Dickson, J. McAllister, and J. McCanny, ''QR decomposition-based matrix inversion for high performance embedded MIMO receivers,'' *IEEE Trans. Signal Process.*, vol. 59, no. 4, pp. 1858–1867, Apr. 2011.

[21] V. Vijayan and K. F. K. Jiavana, ''Implementation of QR decomposition for MIMO detection,'' *Int. J. Eng. Res. Technol.*, vol. 4, no. 4, Apr. 2015, doi: 10.17577/IJERTV4IS040480.

[22] F. Merchant, A. Chattopadhyay, G. Garga, S. K. Nandy, R. Narayan, and N. Gopalan, ''Efficient QR decomposition using low complexity column-wise givens rotation (CGR),'' in *Proc. 27th Int. Conf. VLSI Design 13th Int. Conf. Embedded Syst.*, Mumbai, India, Jan. 2014, pp. 258–263.

[23] J. Gunnels, C. Lin, G. Morrow, and R. van de Geijn, ''A flexible class of parallel matrix multiplication algorithms,'' in *Proc. 1st Merged Int. Parallel Process. Symp. Symp. Parallel Distrib. Process.*, 1998, pp. 110–116.

[24] F. Merchant, T. Vatwani, A. Chattopadhyay, S. Raha, S. K. Nandy, and R. Narayan, ''Efficient realization of householder transform through algorithm-architecture co-design for acceleration of QR factorization,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 8, pp. 1707–1720, Aug. 2018.

[25] F. Merchant, T. Vatwani, A. Chattopadhyay, S. Raha, S. K. Nandy, R. Narayan, and R. Leupers, ''Efficient realization of givens rotation through algorithm-architecture co-design for acceleration of QR factorization,'' 2018, *arXiv:1803.05320*. [Online]. Available: http://arxiv.org/abs/1803.05320

[26] F. Merchant, T. Vatwani, A. Chattopadhyay, S. Raha, S. Nandy, R. Narayan, and R. Leupers, ''A systematic approach for acceleration of matrix-vector operations in CGRA through algorithm-architecture co-design,'' in *Proc. 32nd Int. Conf. VLSI Design 18th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2019, pp. 64–69.

[27] R. Diagle, ''Vantage sequence programming tutorial,'' NDTE Int., Verasonics, Inc., Kirkland, WA, USA, Tech. Rep., Jan. 2017.

[28] J. Liu, ''DCD algorithm: Architectures, FPGA implementations and applications,'' Ph.D. dissertation, Commun. Res. Group Dept. Electron., Univ. York, York, U.K., 2008.

[29] N. Hema, J. U. Kidav, and B. Lakshmi, ''VLSI architecture for broadband MVDR beamformer,'' *Indian J. Sci. Technol.*, vol. 8, no. 19, pp. 1–10, Aug. 2015.

[30] S.-S. Wang, Y.-C. Tien, Y.-T. Hwang, J.-F. Lin, and G.-Z. Wu, ''MVDR based adaptive beamformer design and its FPGA implementation for ultrasonic imaging,'' in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Jeju, South Korea, Oct. 2016, pp. 143–145.

[31] J. U. Kidav, M. N. M. Siva, and M. P. Perumal, ''A parallel complex divider architecture based on DCD iterations for computing complex division in MVDR beamformer,'' *J. Syst. Eng. Electron.*, vol. 29, no. 6, pp. 1124–1135, Dec. 2018.

[32] D. Wang, P. J. Ren, and L. B. Liu, ''A high-throughput fixed-point complex divider for FPGAs,'' *IEICE Electron. Express*, vol. 10, no. 4, pp. 1–8, 2013.

[33] F. Edman and V. Owall, ''Fixed-point implementation of a robust complex valued divider architecture,'' in *Proc. Eur. Conf. Circuit Theory Design*, Sep. 2005, pp. 143–146.

[34] J. Liu, B. Weaver, and Y. Zakharov, ''FPGA implementation of multiplication-free complex division,'' *Electron. Lett.*, vol. 44, no. 2, pp. 95–96, Jan. 2008.

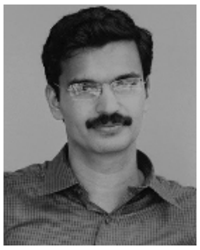
[35] S. Qiao, ''Hybrid algorithm for fast Toeplitz orthogonalization,'' *Numerische Math.*, vol. 53, pp. 351–366, May 1988.

[36] *LogiCORE IP Fast Fourier Transform V 7.1*, Xilinx, San Jose, CA, USA, Mar. 2011.

[37] J. Chen, A. C. H. Yu, and H. K.-H. So, ''Design considerations of real-time adaptive beamformer for medical ultrasound research using FPGA and GPU,'' in *Proc. Int. Conf. Field-Program. Technol.*, Dec. 2012, pp. 198–205.

[38] J. Liu, B. Weaver, Y. Zakharov, and G. White, ''An FPGA-based MVDR beamformer using dichotomous coordinate descent iterations,'' in *Proc. IEEE Int. Conf. Commun.*, Jun. 2007, pp. 2551–2556.

[39] J. Liu, Z. Quan, and Y. Zakharov, ''Parallel FPGA implementation of DCD algorithm,'' in *Proc. 15th Int. Conf. Digit. Signal Process.*, Cardiff, U.K., Jul. 2007, pp. 331–334



S. G. SREEJEESH received the Bachelor of Technology degree (Hons.) in electronics and communication engineering (ECE) from Kannur University, India, in 2002. He is currently pursuing the M.Tech. degree (By Research) with the Vellore Institute of Technology (A deemed to be university), Vellore. He joined as a Research Trainee with NTT Basic Research Laboratories, Japan, in 2003, where he has worked in device characterization of carbon nanotube devices. From 2004 to 2006, he has worked as an Electronics Engineer with INEL Electronics, India and United Arab Emirates. He is currently working as a Senior Technical Officer with the National Institute of Electronics and Information Technology (NIELIT) Calicut, Ministry of Electronics and Information Technology (MeitY), Government of India. His research interests include implementations of high-performance VLSI signal processing architecture for biomedical applications on FPGA and ASIC.

R. SAKTHIVEL (Senior Member, IEEE) received the bachelor's degree in electrical engineering from Madras University, in 2000, the M.E. degree in applied electronics from Anna University, in 2004, and the Ph.D. degree in the area of low-power high-speed architecture development for signal processing and cryptography. He is currently working as an Associate Professor with the School of Electronics Engineering, Vellore Institute of Technology, Vellore. He is a coauthor of the *Basic Electrical Engineering* (Sonaversity, 2001) and the author of the *VLSI Design* (S. Chand, 2007). He has published more than 60 peer reviewed papers in international conferences/journals. He has delivered around 50 guest lectures/invited talk and hands on workshop in the area of FPGA-based system design, full custom IC design, RTL to GDSII, and ASIC Design. His research interests include low-power VLSI design, developing high-speed architecture for cryptography, and image processing. His current research interests include neuromorphic computing and making efficient hardware for AI and ML.

JAYARAJ U. KIDAV received the Bachelor of Engineering degree (Hons.) in electronics and communication engineering from Madurai Kamaraj University, India, in 2000, the Master of Engineering degree in VLSI design from the PSG College of Technology, Bharathiar University, India, in 2002, and the Ph.D. degree from the Karunya Institute of Technology and Sciences (A deemed to be university), Coimbatore, India, in 2020. He joined as the Scientist of the Defense Research and Development Organization (DRDO), Ministry of Defense, Government of India, in 2002, where he has worked in signal processing systems development for defense applications. From 2008 to 2010, he has worked as a Research and Development Engineer with the Systems and Technology Group, IBM India Pvt., Ltd. He is currently working as the Scientist of the National Institute of Electronics and Information Technology (NIELIT) Calicut, Ministry of Electronics and Information Technology, Government of India. His research interests include high-performance VLSI signal processing architecture development for adaptive beamformer in high-sampling rate applications like medical ultrasound imaging and RADAR.