An FPGA Implementation of Givens Rotation Based Digital Architecture for Computing Eigenvalues of Asymmetric Matrix

Ilayda Koseoglu[1], Elif Ozturk[1], Tuba Ayhan[2], and Mustak E. Yalcin[1]

[1]Department of Electronics & Communication Engineering Istanbul Technical University, Istanbul, Turkey
ilaydakoseoglu98@hotmail.com, elifozturk1109@gmail.com, mustak.yalcin@itu.edu.tr
[2]Electrical and Electronics Eng. Dept. MEF University Istanbul, Turkey
ayhant@mef.edu.tr

Abstract

This paper proposes the digital circuit design that performs the eigenvalue calculation of asymmetric matrices with real-valued elements. Eigenvalues are computed iteratively through the QR algorithm. In the QR algorithm, the input matrix is factorized into orthogonal Q and upper triangular R matrix, then the RQ product is calculated to obtain an iterated matrix. For a time-efficient QR decomposition process, the Givens Rotation (GR) Principle is utilized to benefit from the parallelization feature. Parallelization is managed by the Systolic Array (SA) architecture that is created by placing Givens Generation (GG) and Row Updates (RU) blocks in a triangle array. In this paper, 4x4 input matrix is used to create a TSA architecture including n-1 diagonal (GG), and (n * (n-1)) / 2 off-diagonal (RU) modules. In the results section, Givens Rotation is compared with the Gram Schmidt algorithm used in our previous study [1] in terms of error, and area usage.

1. Introduction

Eigenvalue calculation is fundamental in the communication technologies and digital signal processing fields. One of the main concerns in these fields is the speed factor. Since eigenvalue calculations are handled with iterative methods, it causes time loss. Digital design is implemented having parallelization concerns to provide a time-efficient system. The parallelization is achieved by using the Systolic Array.

Systolic Array (SA) [2] architecture is a digital architecture that allows parallelization to obtain a time-efficient system. However, it becomes inefficient in terms of area usage to save time. In addition to time efficiency, it also provides ease of design as it has a simple and modular structure.

As mentioned in the previous study [1], QR Algorithm [3] is chosen to work on asymmetric matrices. As a difference, the QR Decomposition structure in QR Algorithm is implemented using the Givens Rotation Principle [4] instead of Gram Schmidt Orthogonalization [5]. In both studies, the design is made using only basic mathematical modules and square root modules. The same submodules are used to make an accurate comparison of the two algorithms. In the literature, there are different designs of these algorithms. Especially for Givens Rotation, Coordinate Rotation Digital Computer (CORDIC) cores are frequently used.

Theoretical information about the QR algorithm and GR Principle is explained in the second chapter. Then, the implementation of TSA architecture to digital design is explained in the third chapter. Subsequently, in the fourth chapter, the implementation results are compared for GR and MGS. In the end, the conclusion is achieved.

2. QR Algorithm with Givens Rotation Principle

2.1. QR Algorithm

The QR algorithm contains a GRD block, in which the matrix is decomposed into Q orthogonal and R upper triangular matrices. Then the matrix is updated at each iteration by performing the RQ matrix multiplication. This process ends when the lower diagonal elements are approaching zero. When concluded, the diagonal elements form the eigenvalues. However, this algorithm is modified for complex results and is given in Algorithm 1 [1, 3].

Algorithm 1: QR algorithm (Modified) [1, 3]

input : Matrix A
output : Eigenvalues = Diagonal elements of A_{out}

while ~ *Lower Diagonal elements starting from (n+2,n) of A_i approach to zero* **do**
  [Q, R] ← QR Decomposition(A_i);
  i ← i+1;
  A_i ← R*Q;
end while
while ~ *Lower Diagonal elements (n+1,n) of A_i not equal to zero* **do**
  Extracted Matrix = [(n,n), (n+1,n); (n,n+1),(n+1,n+1)] ← Extract the 2x2 matrix from A_i;
  [eig_reel + eig_complex; eig_reel - eig_complex] ← Complex Conjugate Pair Calculation Block (Extracting Matrix)
  n ← n + 1;
end while

2.2. QR Decomposition

Givens Rotation principle that forms a plane rotation spanned by two coordinate axes. With this principle, the lower triangular part of the original matrix is brought closer to zero at each iteration.

The process [4]:

1. Starting from the first column of an n by n asymmetric matrix, the first non-zero lower triangular element is determined to make that element zero. For this, an n by n Givens matrix should be formed with a subsection in which its -sin θ element is put at the same row and column number of the previously determined matrix element. Similarly, cos θ is placed as the first diagonal element of the column. The other diagonals of Givens

are always defined as one, and all other matrix elements become zero. Then this G matrix is multiplied with the original matrix from the left. The resultant matrix will have a zero matrix element in the previously determined spot. This procedure is repeated until the first column elements are zeroed through the updated matrix instead of the original one.

2. The first process is repeated for all other columns until the whole matrix is checked.

3. The last obtained matrix becomes the R of the decomposition. To find the Q matrix determined G matrices are multiplied respectively from the left. Then the transpose of this multiplication is calculated.

4. To calculate the transpose, all diagonal elements of the matrix are kept, and the other matrix elements are changed in pairs by switching each element's row and column values. In this way, Q of the decomposition is achieved. This matrix is called the plane rotator acting in the plane of row and column values determined in the first process.

Algorithm 2: Givens Rotation Principle [4]

```
input  : Matrix A
output : Upper Diagonal R and Orthogonal Q Matrices
for j = 1:n do
  for i = 1:n do
    if i > j do
      if R(i , j) ~= 0 do
        cos← R(j,j)/(sqrt( (R(j,j)*R(j,j)) +  (R(i,j)*R(i,j)) ));
        sin← R(i,j)/(sqrt( (R(j,j)*R(j,j)) +  (R(i,j)*R(i,j)) ));
        G(j,j)← cos;
        G(i,i)← cos;
        G(j,i)← sin;
        G(i,j)← -sin;
        R← G*R;
        Q← G*Q;
      end if
    end if
  end for
end for
```

3. Hardware Architecture

QR Algorithm given in Algorithm 1 is designed for digital architecture by creating separate blocks for fundamental units. The QR Decomposition block is given the input matrix, then the output Q and R matrices enter the Matrix Multiplication block for RQ multiplication. These two basic modules are sufficient for the iteration cycle, so it is checked at this stage that the matrix approaches zero. If not zero, the updated matrix is returned to the QRD block. When the iterations are completed, it is rechecked in the Complex Conjugate Check block. If it contains complex results, complex eigenvalue calculations are made within the Eigenvalue Calculation For The 2x2 block.

3.1. QR Decomposition on Triangular SA Architecture

In the QRD block design, the GR algorithm is implemented using the Triangular SA (TSA) structure. A triangular structure is shown in Figure 2 is created by combining IC and BC. However, the BC and IC modules in this architecture are different for each column. Area usage is reduced by minimizing the circuit in each column. In the BC module, the sin and cos values, which are the elements of the givens rotations matrix, are produced, so it is also called the Givens Generation (GG). In IC modules, the rows of the matrix are updated and called Row Update (RU). In each iteration, the diagonal element R is the output signal from module BC. R upper diagonal elements are the output signal from the IC module. In this structure, n - 1 diagonal (GG) modules, (n (n-1)) / 2 off-diagonal (RU) modules are used.

The GR algorithm given in Algorithm 2 includes a nested for loop. But all calculations are done under the inner for loop. That indicates all operations must be done for each lower diagonal element of the input matrix, and thus the architecture cannot be parallelized enough. Therefore, Column-wise Givens Rotation (CGR) is used to adapt the algorithm to the TSA structure. The sin and cos values of the Givens rotation matrix are calculated for each column instead of each lower diagonal element. Thus, the BC module is created. These sin, cos, k, and L values are then given to the RU modules in the same TSA row to update the R matrix. Updates of the R matrix are performed within the RU module. Then the updates create the new matrix.

From the n x n input matrix, (n)x(n-i) sized intermediate matrix is obtained in i^{th} iteration. The 1^{st} row of this matrix forms the i^{th} row of the R output matrix. The remaining (n-i) x (n-i) dimensional part of the intermediate matrix enters the next iteration as the updated matrix.

For a 4x4 matrix example matrix, p3 (norm of columns) from GG1 block output and updates obtained from the RU1 block are placed as the first row. At the same time, the elements below the first element p3 in Column1 are zeroed. Calculated Row1 and Column1 form the first column and row of the output matrix R. Remaining matrix elements of the latest matrix are obtained by updating the input matrix columns using the sin and cos values with the operations shown in Figure 1. These operations are performed by taking the first column and the other columns two by two in the RU1 block. At the same time, the 3x3 part of the updated matrix, excluding the first row and first column, creates the new matrix. With this new matrix, the same operations are repeated similarly to obtain a 2x2 matrix. The new values received at this stage form the final elements of the R matrix.

$$GX = \begin{bmatrix} p_3 & \frac{x_{11}x_{12}+s_{11}}{p_3} & \frac{x_{11}x_{13}+s_{12}}{p_3} & \frac{x_{11}x_{14}+s_{13}}{p_3} \\ 0 & \frac{x_{11}s_{11}}{p_3} - \frac{x_{12}p_2}{p_3} & \frac{x_{11}s_{12}}{p_3} - \frac{x_{13}p_2}{p_3} & \frac{x_{11}s_{13}}{p_3} - \frac{x_{14}p_2}{p_3} \\ 0 & \frac{x_{21}s_{21}}{p_3}\frac{p_3p_2}{} - \frac{x_{22}p_1}{p_2} & \frac{x_{21}s_{22}}{p_3}\frac{p_3p_2}{} - \frac{x_{23}p_1}{p_2} & \frac{x_{21}s_{23}}{p_3}\frac{p_3p_2}{} - \frac{x_{24}p_1}{p_2} \\ 0 & \frac{x_{31}x_{42}}{p_1}\frac{p_2p_1}{} - \frac{x_{41}x_{32}}{p_1} & \frac{x_{31}x_{43}}{p_1}\frac{p_2p_1}{} - \frac{x_{41}x_{33}}{p_1} & \frac{x_{31}x_{44}}{p_1}\frac{p_2p_1}{} - \frac{x_{41}x_{34}}{p_1} \end{bmatrix}$$

$$= \begin{bmatrix} p_3 & \frac{x_{11}x_{12}+s_{11}}{p_3} & \frac{x_{11}x_{13}+s_{12}}{p_3} & \frac{x_{11}x_{14}+s_{13}}{p_3} \\ 0 & k_1s_{11} - x_{12}l_1 & k_1s_{12} - x_{13}l_1 & k_1s_{13} - x_{14}l_1 \\ 0 & k_2s_{21} - x_{22}l_2 & k_2s_{22} - x_{23}l_2 & k_2s_{23} - x_{24}l_2 \\ 0 & cx_{42} - x_{32}s & cx_{43} - x_{33}s & cx_{44} - x_{34}s \end{bmatrix}$$

Fig. 1. Updated matrix of 4x4 input matrix after one iteration [4].

GG (Figure 3) includes multiplication, addition, division, and square root operations. The input x is the i^{th} row and j^{th} column element of the input matrix. The output p3 in GG1 forms the first diagonal element of the R matrix. The output p2 signal in the GG2 block and p1 signal in the GG3 block form the second and third diagonal elements of the R matrix, respectively. The fourth diagonal element is the row4 update output of the RU3 module. L, k outputs, and sin and cos outputs of Givens Rotation matrix are produced to be used in RU modules. RU

(Figure 4) includes all basic arithmetic operations. Row updates, which are the outputs of RU, forms the input matrix of the new iteration. Arithmetic IPs in Vivado are used for arithmetic operations in cells. Then the sign errors are prevented by adding two's complements to the beginning of the multiplication and division modules.

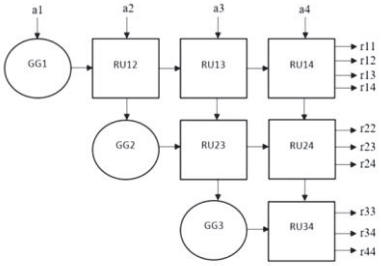
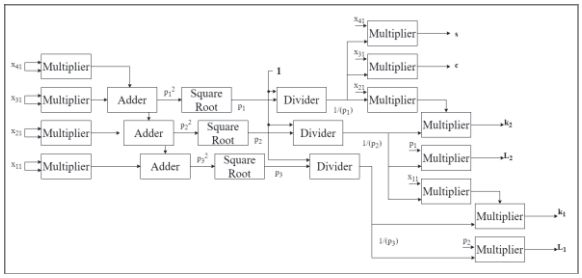
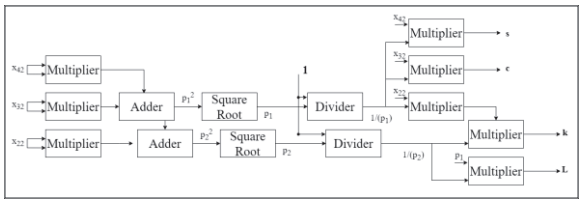


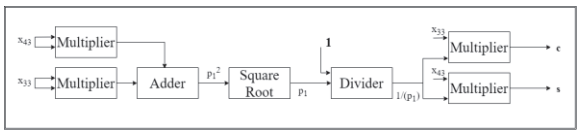
Fig. 2. Triangular Systolic Array for Givens Rotations.



(a)

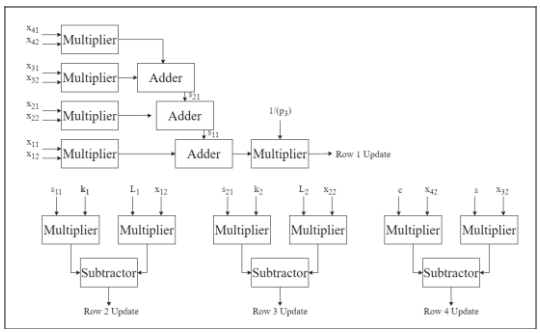


(b)

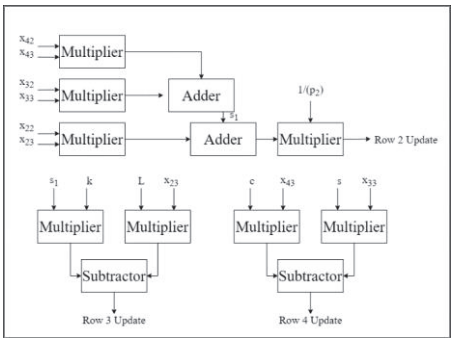


(c)

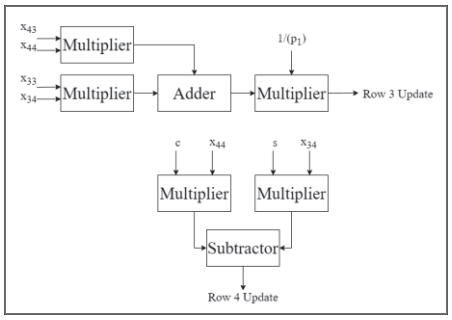
Fig. 3. a) GG1 module b) GG2 module. c) GG3 module [4].



(a)



(b)



(c)

Fig. 4. a) RU1 module. b) RU2 module. c) RU3 module [4].

The square root operation in the modules is customized based on the requirements of the study it is explained in detail in our previous work [1, 6].

With the TSA structure, only the R matrix is calculated. Then, the Q matrix is obtained by using the A=QR equation for eigenvalue calculation.

The 1st column is obtained as

$$q_{i1}=a_{i1}r_{11}. \quad (1)$$

The 2nd column is obtained as

$$q_{i2}=a_{i2}-q_{i1}r_{12}r_{22}. \quad (2)$$

The 3rd column is obtained as

$$q_{i3}=a_{i3}-q_{i1}r_{13}-q_{i2}r_{23}r_{33}. \quad (3)$$

The 4th column is obtained as

$$q_{i4}=a_{i4}-q_{i1}r_{14}-q_{i2}r_{24}-q_{i3}r_{34}r_{44}. \quad (4)$$

While the same operations are performed in parallel for each row, the elements in the same row are calculated sequentially. Thus, the square SA structure is formed (Figure 5).

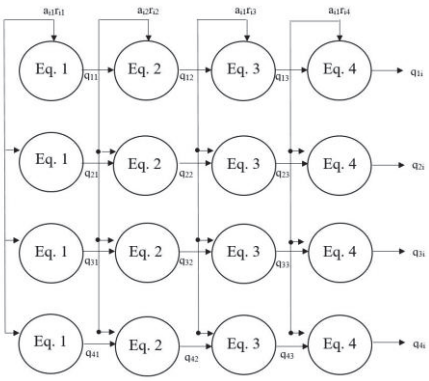


Fig. 5. SA structure for Q estimation block.

3.2. Matrix Multiplication Block

In the Matrix Multiplication block, the i^{th} row of R and the j^{th} column of Q are multiplied to get the ij^{th} element of the result matrix. Since the maximum parallelization in this block is aimed, the use of the area is traded.

3.3. Diagonality and Complex Conjugate Pairs Check Blocks

Lower diagonal elements starting from (n+2,n) of A matrix should approach zero checked in the Diagonality Check block. And iteration continues thanks to this block until these elements are zero. When the iterations are completed, the diagonal elements of the matrix are drawn, and the eigenvalues are obtained. Then the matrix enters the Complex Conjugate Pairs Check block, and it is checked whether $((i + 1), i)^{th}$ elements are zero in this block. If any of them is not zero, it is understood to contain a complex conjugate pair. This stage is not included in the QR Algorithm, so the algorithm is modified. For the non-zero (n+1,n) element, a 2x2 extracted matrix is obtained: Extracted Matrix = [(n,n), (n+1,n); (n,n+1),(n+1,n+1)].

3.4. Eigenvalue Calculation for 2x2 Matrix Block with Solving Quadratıc Equation Architecture

Eigenvalue Calculation for 2x2 Matrix Block takes the extracted matrix as the input matrix and sets the determinant of this matrix to zero, then solves it as a quadratic equation. The roots of the equation give the eigenvalues [1].

4. Implementation Results

In this chapter, behavioral simulation and post-synthesis reports are analyzed. In Figure 6, eigenvalue calculation behavioral simulation is given. When the signals are examined, the X matrix represents the columns of the input matrix. The signals below are the elements of the R matrix and the columns of the Q matrix. The exit_flag signal just below becomes active at the end of 3 iterations and stops the iteration. The [255:0] size signal in the orange box is the 4x4 final update matrix. After this matrix is checked in the Complex Conjugate Check block, it is understood that it contains a complex result, and an extracted matrix is created. Finally, the Complex conjugate pair signals eig_comp and eig_reel, which are the results of non-zero parts, and the diagonals of the final updated matrix form the eigenvalues. This simulation is run at 100 MHz, and the results are obtained at approximately 25 us.

When the simulation is compared with the previous work [1], there are fewer iterations in GR simulation and therefore the results come in a shorter time. However, this does not imply that GR is faster since a GR iteration takes about 80 clock cycles, while an MGS takes about 60 clock cycles. However, since an equal amount of time is determined for each iteration in the top module FSM, GR, having fewer iteration counts, achieved results faster. The simulations use a 100 MHz clock frequency. That's why in the MGS simulation, it takes 330 clock cycles to get the result. However, if the iteration number is multiplied by the time required for an iteration, 240 clock cycles are found. When combined with these other modules, 250 clock cycles can be found, which is close to the output time value of GR.

Examining the post-implementation utilization report (Xilinx Vivado® Design Suite 2018.2 with Kintex-7), the design has 122976 Slice LUT and 88599 Slice Register resources.

In Table 1, the operations used for the n by n input matrix are given and compared. The number of operations is given depending on the variable n. Looking at the differences, MGS [1] uses fewer operations from 8 onwards, and more multiplication and division modules are used when n is less than 7. For example, for a 4x4 matrix, the MGS module uses 64 multiplication, 24 addition, 24 subtraction, 28 division, and 4 square root modules. GR uses 48 multiplications, 12 additions, 30 subtractions, 22 divisions, and 6 square roots. Thus, it is seen that operations other than square root and subtraction are used more in MGS. The main reason for this is that the internal structures of the BC and IC modules of the GR algorithm are getting smaller and smaller.

$Input\ matrix$: [1, 1.5, 1, 1.5; 1.5, 1, 1.5, 1; 1, 1, 1.5, 1.5; 1.5, 1.5, 1, 1.5]

The eigenvalues of the input matrix are shown in Table 2. These values are compared with MATLAB outputs to determine the error amounts. For this specific example input matrix, the GR eigenvalue outputs are calculated closer to MATLAB results when compared with MGS [1] error results.

5. Conclusions

The paper aims to design a digital system that calculates eigenvalues of asymmetric matrices with the Givens Rotation-based QR algorithm. As a result, by combining and customizing the studies in the literature, the QR Algorithm was implemented with two different QRD blocks: Givens Rotation in this paper and Modified Gram Schmidt in the previous [1]. Both designs have been implemented with TSA architectural structure, including only mathematical and square root operations, thus creating an environment for comparing two algorithm design implementations. Based on the example given in the study, although Givens Rotation has more area usage, the algorithm reached the result faster and with more accurate values. Gram Schmidt algorithm, on the other hand, has a more modular architecture to implement in TSA. In addition, the comparison results may vary according to the matrix size. Another significant point of the study is the targeting of a time-effective system. For this, parallelization with TSA architecture is made. On future optimization, studies may be a focus in terms of space usage and working speed. Time efficiency can be achieved by increasing the frequency speed. The basic operations used by the modules can be custom-designed instead of using IP cores.

6. Acknowledgement

This work was supported by the project with the name “Instruction Extension of RISC-V Processor for Driver Fatigue Detection system and Implementation”, carried with the number 119N641 under the Cooperation Agreement between the Scientific and Technological Research Council of Turkey (TUBITAK) International 2535 and Iran Ministry of Science, Research and Technology (MSRT).

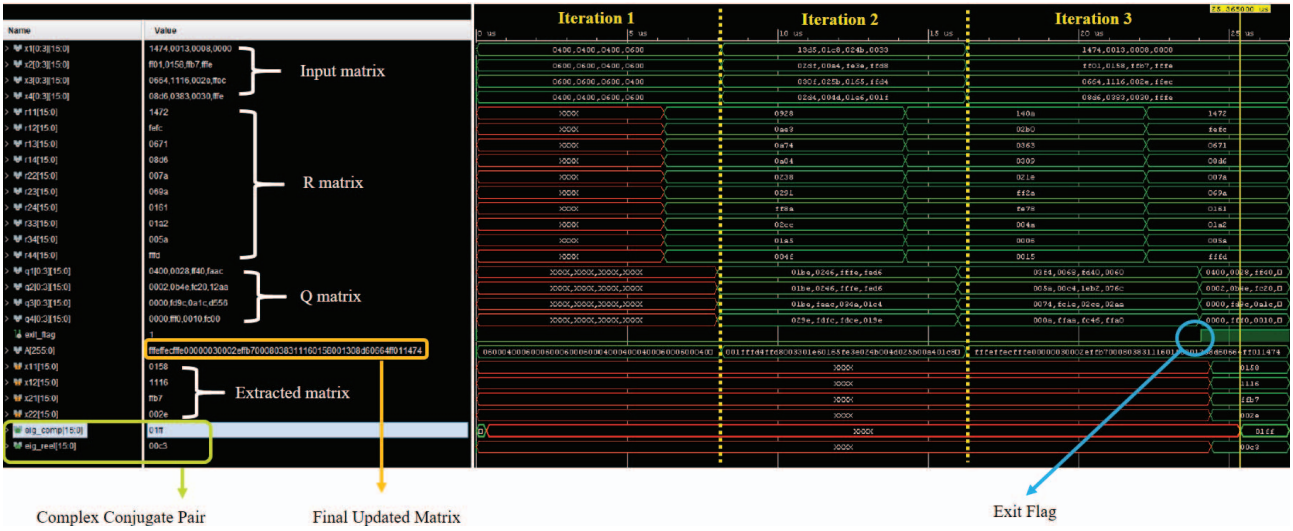


Fig. 6. Simulation of Givens Rotation based QR Algorithm.

Table 1. Number of operations for n by n input matrix.

Operation	Givens Rotation	Modified Gram Schmidt	Matrix Multiplication Block	Quadratic Equation Solver	Total for GR	Total for MGS	Differences (GR - MGS)
Multiplication	$9n^2/2-7n/2-2$	$4n^2$	$4n^2$	3	$17n^2/2 - 7n/2 +1$	$8n^2+3$	$n^2/2 - 7n/2 -2$
Addition	n^2-n	n^2+2n	$3n^2$	1	$4n^2-n+1$	$4n^2+2n+1$	-3n
Subtraction	$5(n^2-n)/2$	$2n^2-2n$	-	2	$5(n^2-n)/2 + 2$	$2n^2-2n+2$	$(n^2-n)/2$
Division	$3n^2/2 - n/2$	n^2+3n	-	-	$3n^2/2 - n/2$	$3n^2/2 - n/2$	$n^2/2-7n/2$
Square Root	$(n^2-n)/2$	n	-	1	$(n^2-n)/2 +1$	n + 1	$n^2/2-3n/2$
Shifter	-	-	-	3	3	3	-
2's Complement	-	-	-	2	2	2	-

Table 2. Eigenvalue comparison with MATLAB for 4x4 matrix.

Eigenvalues	MATLAB Implemented QR Algorithm with GR	MATLAB Implemented QR Algorithm with MGS	Digital Implemented QR Algorithm with GR	Digital Implemented QR Algorithm with MGS	Error for GR	Error for MGS
Eigenvalue 1	5.1227 + 0.0000i	5.1227 + 0.0000i	5.1132 + 0.0000i	5.1113 + 0.0000i	0.0095 + 0.0000i	0.0114+ 0.0000i
Eigenvalue 2	0.1887 + 0.5307i	0.1887 + 0.5307i	0.1904 + 0.4990i	0.1826 + 0.4990i	-0.0017 + 0.0317i	0.0060+ 0.0317i
Eigenvalue 3	0.1887 - 0.5307i	0.1887 - 0.5307i	0.1904 - 0.4990i	0.1826 - 0.4990i	-0.0017 - 0.0317i	0.0060 - 0.0317i
Eigenvalue 4	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0009 + 0.0000i	0.0000 + 0.0000i	-0.0009 + 0.0000i	0.0000+ 0.0000i

7. References

[1] E. Ozturk, I Koseoglu, and M.E. Yalcin "A Triangular Systolic Array Based Digital Architecture for Computing Eigenvalues of Asymmetric Matrix," The 17th IEEE International Workshop on Cellular Nanoscale Networks and their Applications (CNNA), 2021.

[2] H.T. Kung, "Why Systolic Architectures?," IEEE Computer, Vol. C-31, pp. 37–46, 1982.

[3] S. S. Dubey, R. Shrestha and S. R. Chowdhury, "A novel architecture for computing eigenvalues of matrix for high speed applications," 2016 IEEE Annual India Conference (INDICON), 2016, pp. 1-5, doi: 10.1109/INDICON.2016.7838959.

[4] V. Vijayan, & K.F. Jiavana, "Implementation of QR Decomposition for MIMO Detection," International Journal of Engineering Research & Technology (IJERT) vol. 4, 2015.

[5] A. Alhamed and S. Alshebeili, "FPGA implementation of complex-valued QR decomposition," 2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA), 2016, pp. 1-4, doi: 10.1109/ICEDSA.2016.7818557.

[6] R. V. W. Putra, "A novel fixed-point square root algorithm and its digital hardware design," International Conference on ICT for Smart Society, 2013, pp. 1-4, doi: 10.1109/ICTSS.2013.6588110.