Implementation QR decomposition based on triangular systolic array

Safaa S. Omran, Ahmed K. Abdul-abbas*
Electrical Engineering Technical College
Middle Technical University
Baghdad/Iraq

Abstract: QR decomposition is widely used in many fields as data processing, image processing, communication systems, multiple input multiple output (MIMO), radar systems, linear algebra and so on. QR decomposition has been computed by using the Householder transformation, givens rotation and Gram Schmidt, these algorithms are mostly used and basic ways for computing a QR decomposition. This paper presents design of Triangular systolic array processor to perform QR decomposition based on Givens Rotation algorithm for a real matrix *(4 × 4)* by using Verilog HDL language and implemented by using FPGA board type of virtex-7. A 0.22 µs are required to perform QR decomposition with 4.54 M Matrix per second is a throughput.

1. INTRODUCTION

The important problem of matrix that show in many fields or applications as like image processing, signal processing [1], multi- input and multi- output (MIMO) systems [2] and [3], solution of differential equation, wireless communication systems, etc. is that of "solving a set of simultaneous linear equation". The commonly numerical method used to solve this problem is to triangularize the coefficient of matrix and then use back-substitution. There are many of methods to triangularize of a matrix, however the QR decomposition is a basic method to triangularize of a matrix. Let A be an m × n matrix with full column rank. The QR decomposition of A is A = QR, where Q is an m×m orthogonal matrix and R is an m × n upper triangular matrix. There are different algorithms to find this decomposition but the common algorithms are Gram-Schmidt orthogonalization, Householder transformation algorithm and Givens rotation [4].

In the literature review, there are many papers in which QR decomposition have been implemented on Field Programmable Gate Array (FPGA) by using these methods. Dongdong Chen and Mihai Sima proposed a parallel architecture of an QR decomposition systolic array based on the GR algorithm (Givens rotation algorithm) on field programmable gate array, takes on the direct mapping by 21-fixed-point CORDIC based process units (PU) that can compute the QRD for a 4 × 4 real matrices [5]. Semih Aslan, et.al. proposed another work to compute QRD, they developed the architecture for QRD using the GR algorithm, based on Coordinate Rotation Digital Computer algorithm (CORDIC) and the fixed point calculations, are optimized for FPGA platforms and the design can run at 246MHz as maximum speed [6]. High Throughput hardware design has been proposed by Sergio D. Munoz and Javier Hormigo, utilizing GR algorithm. They used a new two dimensional systolic-array architecture with pipelined-processing

components. which are based on the CORDIC algorithm where they propose to use two dimensional array architecture, where each processing element works with all the elements of the same row and these processing elements are pipelining [7]. while Ali Umut Irturk proposed a different design, a processor architecture has been proposed to perform QRD process. This processor architecture consists of two units to perform addition and subtraction operations, two units to perform multiplication and division operations and one unit to perform square root operation by using newton iteration method [8]. In this paper, QR decomposition was implemented using triangular systolic array based on Givens Rotation algorithm.

The following sections start with general background on Givens Rotation algorithm and its mathematical procedure, and move on to design of 4 by 4 triangular systolic array core is described. Then the implementation result and throughput is described. Finally, the conclusion is given in the last section.

II. GIVENS ROTATION ALGORITHM

The QR decomposition can be computed by using Givens rotation [4]. Givens Rotation algorithm eliminates one element in a matrix one at a time. If matrix A consists of, m x n elements, to compute the QRD by Givens Rotation, the first step, is to use $a_{31}$to eliminate a_{41} and by using equation (1) then, computing Cosθ and Sinθ using equation (2) and (3) respectively.

$$r_{i,j} = \sqrt{a_{ik}^2 + a_{jk}^2} \quad (1)$$

$$\text{Cos}\theta = \frac{a_{ik}}{r_{i,j}} \quad (2)$$

$$\text{Sin}\theta = \frac{a_{jk}}{r_{i,j}} \quad (3)$$

Where: a_{jk}, a_{ik} are elements of matrix A, i = 3, j = 4, k = 1.

Step two, generating G^i matrix (Givens rotation matrix) by I matrix with addition values of Cos and Sin into I matrix. Where $G_{33}^1 = Cos\theta$, $G_{34}^1 = Sin\theta, G_{43}^1 = -Sin\theta$ and $G_{44}^1 = Cos\theta$. As shown on Fig. 1.

$$\mathbf{G}_{3,4}^{(1)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \cos\theta_{3,4} & \sin\theta_{3,4} \\ & & -\sin\theta_{3,4} & \cos\theta_{3,4} \end{pmatrix}$$

Fig.1: $G_{3,4}^1$ matrix.

Step three, computing first rotation using equation (4), when multiplying G^1 and A^0, produce A^1.

$$A^i = G^i * A^{i-1} \tag{4}$$

Then repeating this process five times to compute R matrix. Fig.2 shows these process, and Fig.3 shows the givens rotation algorithm [4].

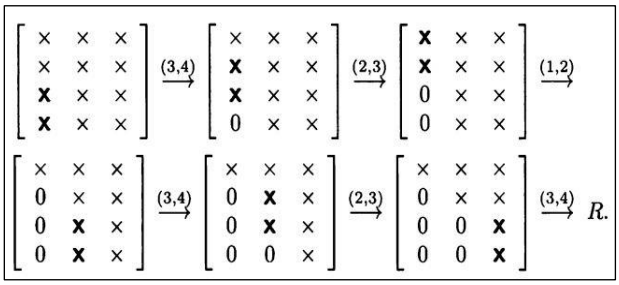

Fig.2: steps to compute R matrix from A (4x3).

```
for j = 1:n
    for i = m: − 1:j + 1
        [c, s] = givens(A(i − 1, j), A(i, j))
        A(i − 1:i, j:n) = [ c  s ; −s  c ]^T A(i − 1:i, j:n)
    end
end
```

Fig.3: Algorithm to find R matrix.

After finding R matrix, equation (5) is used to compute Q (orthogonal matrix) from G matrices. In this step, the problem of matrix multiplication is appearing and this will be explained clearly in section four.

$$Q = (G^i G^{i+1} G^{i+2} \dots \dots G^m)^{-1} \tag{5}$$

III. FPGA IMPLEMENTATION

The methods of matrix factorization such as cholesky decomposition, LU decomposition, Gaussian elimination or QR decomposition can be explained as a set of nested loops with three levels of nesting which it need computational complexity. In another side, the systolic arrays target planar layouts, where it suitable for integrated circuits.

This triangular systolic array consists of two kinds of computational nodes, boundary node (BN) on diagonal of the matrix and internal node (IN) off the diagonal. The BNs are used to compute the Givens rotation that is applied over a specific row in the input matrix. the orthogonal rotation matrix as in equation (6) that can clear one lower triangular element of a decomposed matrix is calculated in the BN, while the rotation parameters C and S are output to the INs. The generated

rotation parameters are sequentially broadcasted to the INs in the same row as the BN from left to right by a specific clock rate. The INs applies this Givens rotation parameters (C, S) received from the BNs on the same row addition to its own input values to calculate new values as outputs. Using the rotation values C and S from the BN, the INs update the remaining elements at the two right rows of the matrix involved in one rotation as particular equation (6).

$$\begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} r_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \quad (6)$$

Fig. 4 shows the block diagram for the 4 by 4 architecture of triangular systolic array, while the Fig. 5 shows the block diagram for the 4 by 4 architecture of boundary node. HDL coder of MATLAB program was used to design the boundary nodes and internal nodes. As clear from Fig. -6, the square root, division, multiplication and addition operations was needed to compute a value of r_n and the givens rotation parameters.

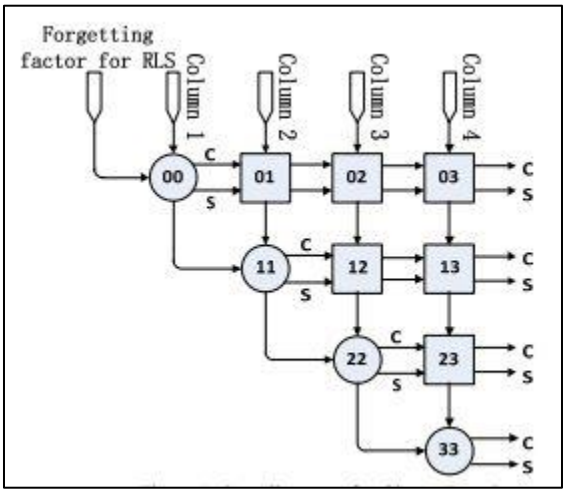

Fig.4: Triangular systolic array.

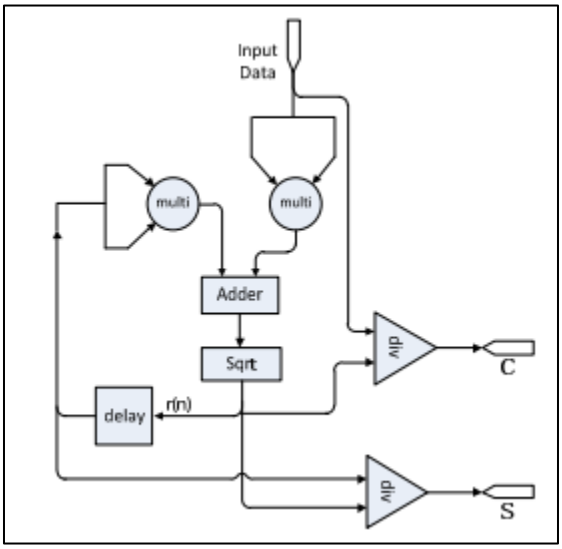

Fig.5: Block diagram of boundary node.

The square root operation is designed by using a bit set shift/addition algorithm, to compute value of square root without addition clock cycles [9]. That means the operation of square root is done in one clock cycle only, so, by this way, the time of execution is reduced for this operation. To use this method, the first step is to select the input port from sources tab and sets the properties (set a type of input port to fixed point (1,32,20), where 1 refers to the sign bit, 32 refers to length of number and 20 refers to length of fraction part), second step is to select the output port from sink tab and then set its properties (set a type of input port to fixed point (1,32,20)). Third step, to select the desired function from math operation, sqrt function and then also set the properties of function (fixed point (1,32,20)). Last step is to generate HDL code from Manu-bar "code." The negative side in this way is the consumption of resources of FPGA kit.

The "Xilinx LogiCORE™ IP Divider" Generator core creates a circuit for integer division based on Radix 2 non -restoring division, or High Radix division with pre-scaling. The Radix 2 algorithm exploits FPGA logic to achieve a range of throughput options that includes single cycle, and the High Radix algorithm exploits DSP slices at lower throughput, but with reuse to reduce resources. The virtex 7 device family support this option, so, Xilinx LogiCORE™ IP Divider was used in this design. Fig. -7 shows the internal block diagram for boundary node.

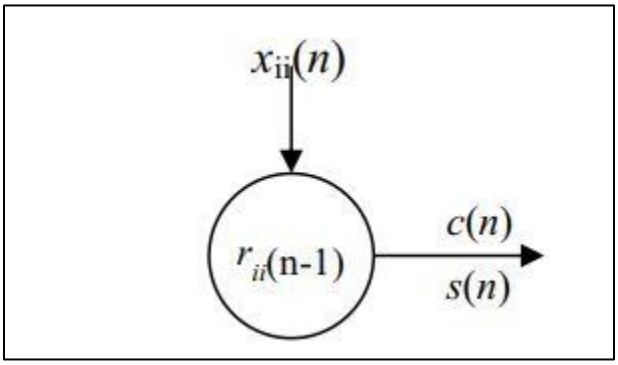

Fig.6: Boundary node.

Where: r(n) = $\sqrt{r^2(n-1) + x^2(n)}$, c(n) = $\frac{r(n-1)}{r(n)}$, s(n) = $\frac{x(n)}{r(n)}$

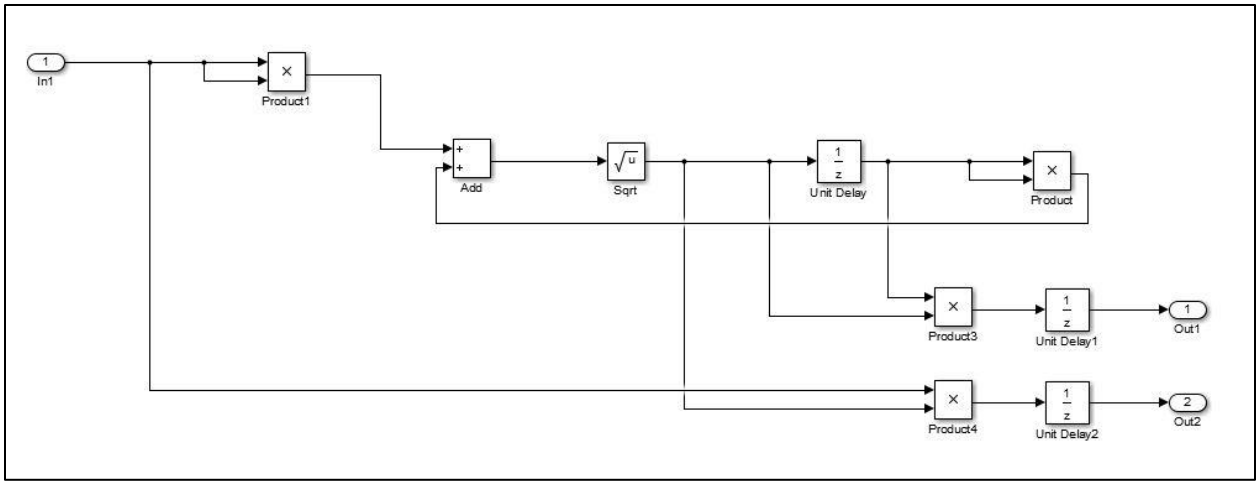

Fig.7: Internal structure boundary node.

While the internal node is simpler than boundary node in hardware design. Where its consisting of several multipliers, adders and delay unit. Fig. -8 shows the block diagram for this node and the Fig. -9 shows the internal structure using HDL coder.

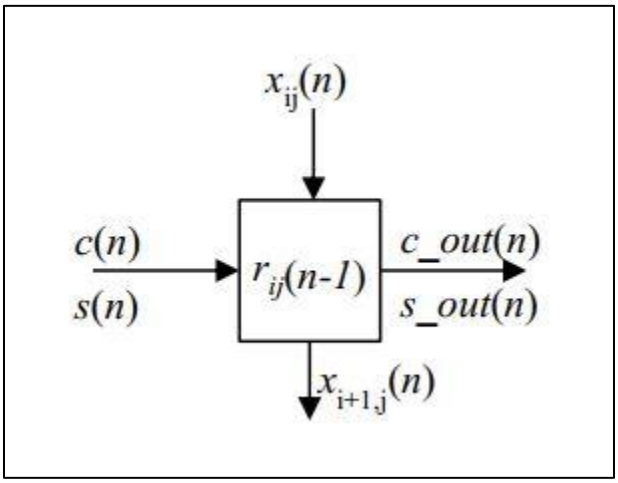

Fig.8: Internal node.

Where: r(n) = c*r(n-1) + s*x(n), x_out = -s*r(n-1) + c*x(n).

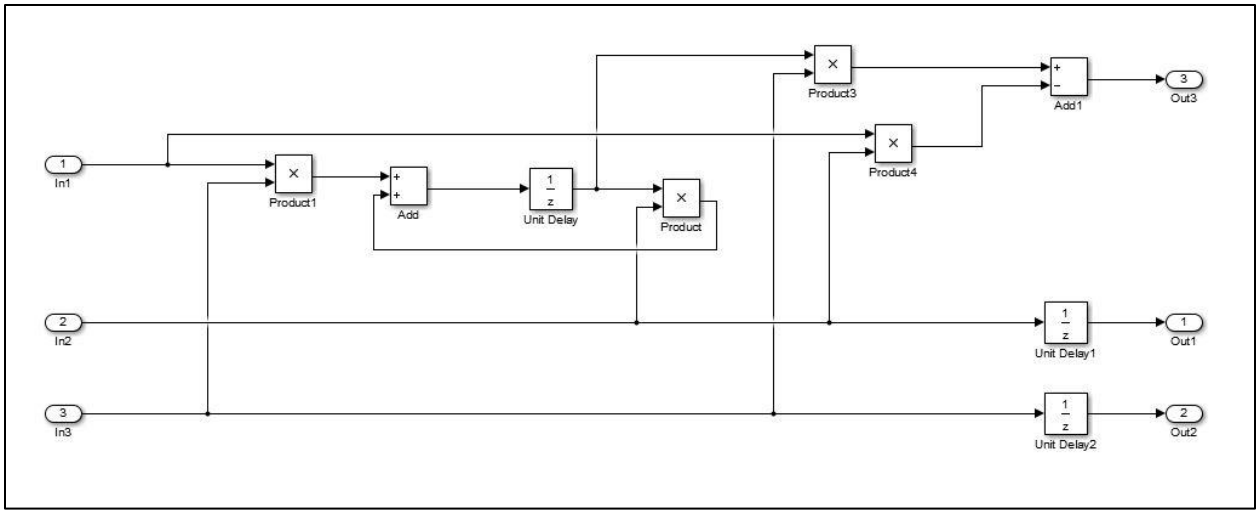

Fig.9: Internal structure internal node.

Last part in section 3, the timing was described for the triangular systolic array. So, assume the time of computation for each node is one clock cycle. As it is shown in table 1 which it is the schedule of the system, the inputs to the system are the rows from top of A matrix that is required to decompose. At time 0 the first element a_{11} is received, the BN 'N_{11}' starts computing, at time 1 the BN 'N_{11}' updates its value of r_{11} and send values of sine and cosine to the IN 'N_{12}', at the same time it starts computing for the second element a_{21}.meanwhile, IN 'N_{12}' receives sine, cosine and a_{12}. It will update the old value of r_{12} and pass new value of r_{12} to the lower level BN 'N_{22}' and passes the values of sine and cosine to the right IN 'N_{13}' at time 3. And so on as shown in table I.

Table I: schedule of triangular systolic array.

time	**BN (input)→(output)**	**IN (input)→(output)**
0	N11{a11}→{c11, s11, update(r11)}at time 1	
1	N11{a21}→{c12, s12, update(r11)}at time 2	N12{a12, c11, s11}→{c11, s11, x122, update(r12)}at time 2
2	N11{a31}→{c13, s13, update(r11)}at time 3 N22{x22}→{c21, s21, update(r22)}at time 3	N12{a22, c12, s12}→{c12, s12, x222, update(r12)}at time 3 N13{a13, c11, s11}→{c11, s11, x123, update(r13)}at time 3
3	N11{a41}→{c14, s14, update(r11)}at time 4 N22{ x222}→{c22, s22, update(r22)}at time 4	N12{a32, c13, s13}→{c13, s13, x322, update(r12)}at time 4 N13{a23, c12, s12}→{c12, s12, x223, update(r13)}at time 4 N14{a14, c11, s11}→{c11, s11, x124, update(r14)}at time 4 N23{ x123, c21, s21}→{c21, s21, x133, update(r23)}at time 4

4	N22{ *x*322}→{*c*23, *s*23, update(*r*22)}at time 5 N33{ *x*133}→{*c*31, *s*31, update(*r*33)}at time 5	N12{*a*42, *c*14, *s*14}→{*c*14, *s*14, *x*422, update(*r*12)}at time 5 N13{*a*33, *c*13, *s*13}→{*c*13, *s*13, *x*323, update(*r*13)}at time 5 N14{*a*24, *c*12, *s*12}→{*c*12, *s*12, *x*224, update(*r*14)}at time 5 N23{ *x*223, *c*22, *s*22}→{*c*22, *s*22, *x*233, update(*r*23)}at time 5 N24{ *x*124, *c*21, *s*21}→{*c*21, *s*21, *x*134, update(*r*24)}at time 5
5	N22{ *x*424}→{*c*22, *s*24, update(*r*22)}at time 6 N33{ *x*233}→{*c*32, *s*32, update(*r*33)}at time 6	N12{*a*52, *c*15, *s*15}→{*c*15, *s*15, *x*522, update(*r*12)}at time 6 N13{*a*43, *c*14, *s*14}→{*c*13, *s*13, *x*423, update(*r*13)}at time 6 N14{*a*34, *c*13, *s*13}→{*c*12, *s*12, *x*324, update(*r*14)}at time 6 N23{ *x*323, *c*23, *s*23}→{*c*23, *s*23, *x*333, update(*r*23)}at time 6 N24{ *x*224, *c*22, *s*22}→{*c*22, *s*22, *x*234, update(*r*24)}at time 6 N34{ *x*134, *c*31, *s*31}→{*c*31, *s*31, *x*144, update(*r*34)}at time 6
6	N33{ *x*333}→{*c*33, *s*33, update(*r*33)}at time 7 N44{*x*144}→{*c*41, *s*41, update(*r*44)}at time 7	N14{*a*44, *c*14, *s*14}→{*c*12, *s*12, *x*424, update(*r*14)}at time 7 N23{ *x*423, *c*24, *s*24}→{*c*24, *s*24, *x*433, update(*r*23)}at time 7 N24{ *x*324, *c*23, *s*23}→{*c*23, *s*23, *x*334, update(*r*24)}at time 7 N34{ *x*234, *c*32, *s*32}→{ *c*32, *s*32, *x*244, update(*r*34)}at time 7
7	N33{ *x*433}→{*c*34, *s*34, update(*r*33)}at time 8 N44{*x*244}→{*c*42, *s*42, update(*r*44)}at time 8	N14{*a*44, *c*15, *s*15}→{*c*12, *s*12, *x*524, update(*r*14)}at time 8 N23{ *x*523, *c*25, *s*25}→{*c*25, *s*25, *x*533, update(*r*23)}at time 8 N24{ *x*424, *c*24, *s*24}→{*c*24, *s*24, *x*434, update(*r*24)}at time 8 N34{ *x*334, *c*33, *s*33}→{*c*33, *s*33, *x*344, update(*r*34)}at time 8
8	N44{*x*344}→{*c*43, *s*43, update(*r*44)}at time 9	N34{ *x*434, *c*34, *s*34}→{ *c*34, *s*34, *x*444, update(*r*34)}at time 9
9	N44{*x*444}→{*c*44, *s*44, update(*r*44)}at time 10	

IV. RESULT OF DESIGN

The proposed design of triangular systolic array has been wrote by using Verilog HDL (Hardware Description Language) and implemented using Virtex- 7 FPGA board. In this design 32-bits fixed-point number representation is used. A testbench is created for testing the results with MATLAB program, The Isim behavior simulation result show that the total latency is 0.22 µs. A MATLAB program generated test matrix is given as input:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

The results as computed by MATLAB program using floating-point representation are:

$$Q = \begin{bmatrix} 0.5774 & 0.2582 & 0.1690 & -0.7559 \\ 0.5774 & 0.2582 & -0.6761 & 0.3780 \\ 0.5774 & -0.5164 & 0.5071 & 0.3780 \\ 0 & 0.7746 & 0.5071 & 0.3780 \end{bmatrix}$$

$$R = \begin{bmatrix} 1.7321 & 1.1547 & 1.1547 & 1.1547 \\ & 1.2910 & 0.5164 & 0.5164 \\ & & 1.1832 & 0.3381 \\ & & & 1.1339 \end{bmatrix}$$

While the results as computed by ISE Design Suite 14.7 program using 32-bits fixed-point representation are:

$$R = \begin{bmatrix} 1.7322 & 1.1548 & 1.1548 & 1.1548 \\ & 1.2911 & 0.5163 & 0.5163 \\ & & 1.1832 & 0.3380 \\ & & & 1.1339 \end{bmatrix}$$

Fig. 10 shows the screen shot of simulation for the result of this design by ISE Design Suite 14.7. The time required for find the QR decomposition of matrix A [4 x 4], was 220 ns which it is clear in Fig. 10. With this time for QR process (0.22 µs), that means the throughput of this design is 4.54 M- Matrix per second. While Fig. 11 shows the device utilization summary.

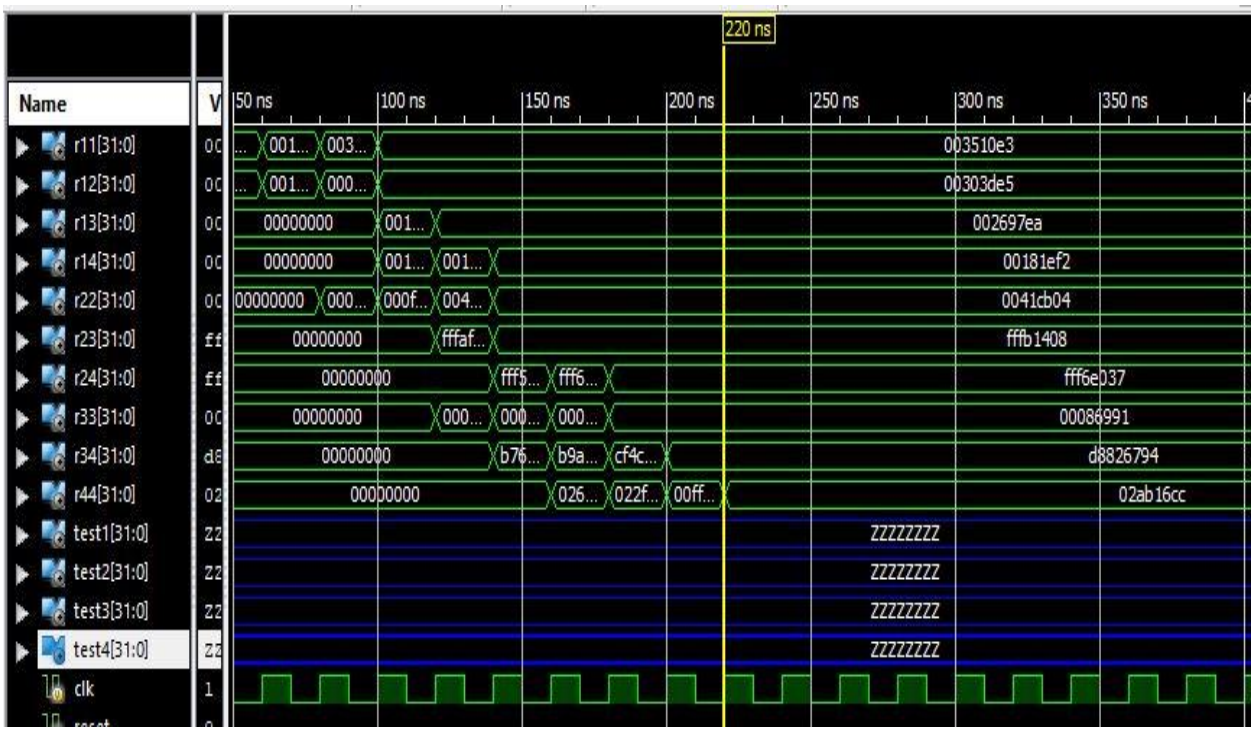

Fig.10: Screen shot for result.

```
Device utilization summary:
---------------------------

Selected Device : 7vx485tffg1761-2


Slice Logic Utilization:
 Number of Slice Registers:             680  out of  607200     0%
 Number of Slice LUTs:                39575  out of  303600    13%
    Number used as Logic:             39575  out of  303600    13%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  40141
   Number with an unused Flip Flop:   39461  out of  40141     98%
   Number with an unused LUT:           566  out of  40141     1%
   Number of fully used LUT-FF pairs:   114  out of  40141     0%
   Number of unique control sets:        10

IO Utilization:
 Number of IOs:                         594
 Number of bonded IOBs:                 460  out of    700    65%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                1  out of     32     3%
 Number of DSP48E1s:                    128  out of   2800     4%
```

Fig.11: device utilization summary.

V. CONCLUSION

In this paper, Triangular systolic array is designed to perform QR decomposition based on Givens Rotation algorithm for a real matrix 4 x 4 elements by using Verilog HDL language. This design is implemented by using FPGA board type of virtex – 7. HDL coder was used to design the boundary node elements and the internal node elements, a 0.22 µs are required to perform QR process without pipelining. And 4.54 M- Matrix per second is a throughput. In future work to enhance this design with pipelining structure.

References

[1] M. Moonen and J. Vandewalle, "A Systolic array for Recursive least square computations," *IEEE Transactions on signal processing*, vol. 41, no. 2, 1993.

[2] P. Luethi, "Gram-Schmidt-based QR Decomposition for MIMO Detection: VLSI Implementation and Comparison," in *IEEE Asia Pacific Conference on Circuits and Systems* , Macao, China , 2008.

[3] Y.-K. K. Y.-J. C. W.-Y. L. a. Y.-S. C. Tsung-Hsien Liu, "Hardware Implementation of the Preprocessing QR-Decomposition for the Soft-Output MIMO Detection with Multiple Tree Traversals," *IEEE Transactions on Circuits and Systems II*, 2016.

[4] G. H. Golub and C. F. V. Loan, Matrix Comutations, Baltimore, Maryland: John Hopkins Univ. Press, 2013.

[5] D. C. a. M. SIMA, "Fixed-Point CORDIC-Based QR Decomposition by Givens Rotations on FPGA," in *Reconfigurable Computing and FPGAs*, Canada, 2011.

[6] S. Aslan, E. Oruklu, and J. Saniie, "Realization of area efficient QR factorization using unified division, square root, and inverse square root hardware," in *IEEE International Conference on Electro/Information Technology*, Canada, 2009.

[7] S. D. M. a. J. Hormigo, "High-Throughput FPGA Implementation of QR Decomposition," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 6, no. 9, p. 1, 2015.

[8] A. U. Irturk, "Implementation of QR Decomposition Algorithm using FPGAs," UNIVERSITY OF CALIFORNIA, CALIFORNIA/Santa Barbara, 2007.

[9] mathwork, "Calculate square root, signed square root, or reciprocal of square root (HDL Coder)," MathWorks, Inc., [Online]. Available: https://www.mathworks.com/help/hdlcoder/ref/sqrt.html.