International Journal of Engineering & Technology, 7 (4) (2018) 2100-2105

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.14517
Research paper

Multi core processor for QR decomposition based on FPGA

Safaa S. Omran [1] *, Ahmed K. Abdul-abbas [1]

[1] *Electrical Engineering Technical College Middle Technical University Baghdad/Iraq*
**Corresponding author E-mail: ahmed89kareem@gmail.com*

Abstract

Hardware design of multicore 32-bits processor is implemented to achieve low latency and high throughput QR decomposition (QRD) based on two algorithms which they are Gram Schmidt (GS) and Givens Rotation (GR). The orthogonal matrices are computed using the first core processor by Gram Schmidt algorithm, and the upper triangular matrices are computed using the second core processor by Givens Rotation algorithm. This design of multicore processor can achieve 50M QRD/s throughput for (4 × 4) matrices at running frequency 200 MHz.

Keywords: *QR Decomposition; Gram Schmidt; Givens Rotation; Multicore Processor; CORDIC Square root.*

1. Introduction

The algebraic matrix operations are very important in the signal processing algorithms. It is used in many fields and the most important of these fields are wireless communication systems, such as beam-forming, multiple input and multiple output (MIMO), cancellation, multi-user detection and so on [1]. The most useful operator in algebraic matrix operations is the QR decomposition, especially for adaptive filtering [2] and MIMO technologies [3]. Some of these systems require high throughput of QR decomposition but they are for small sizes of matrices. The implementation of QR decomposition by hardware is most widely used due to its easy parallelization and its robust numerical properties [4].
Many researchers were interested in QR decomposition based on Field-Programmable Gate-Array (FPGA). P. Luethi [5] proposed a VLSI architecture processor to detect multi-input multi-output (MIMO) up to 1.56 million complex valued of 4x4 dimensional matrices per second. Robert [6] proposed iterative decomposition architecture based on GS algorithm. Ji-Hwan Yoon [7] proposed an architecture to achieve high speed and low latency QR decomposition using Givens Rotation algorithm. Rakesh Gangarajah [8] proposed a parallel hardware architecture to perform QR decomposition using householder transformation algorithm for 5 band carrier aggregation LTE-A (Long Term Evolution-Advanced).
The main objective of this paper is to design and implementation of multi core processor to perform a QR decomposition using two algorithms together which they are Gram Schmidt algorithm and Givens Rotation. In this way the latency is decreased while the throughput is increased.
This paper is organized as follows: in the next section, an overview of QR decomposition with Gram Schmidt algorithm and Givens Rotation algorithm are given. In section 3, an explanation for the design of 32-bits multi core processor. In section 4, the execution and results of the designed processor are given, while in the last section, the conclusion of this work is presented.

2. QR decomposition

A QR-decomposition (also called QR- factorization) of a matrix is the decomposition of the matrix A as the product of A = QR, an orthogonal matrix Q ($QQ^T = I, Q^T = Q^{-1}$) and an upper triangular matrix R (all entries below the main diagonal equal to 0). One note to mention here is that, the QRD of a matrix A is not unique. If A is m × n where m not equal to n, then there are both QR decompositions where Q is either m × m, R is m × n or Q is m × n and R is n × n, where m is the number of rows and n is the number of columns.
Commonly, the Gram Schmidt process, Householder Transformation and Givens Rotations are known as the most widely used algorithms for QR decomposition. The Householder Transformation and givens rotations illustrate the feature of numerical stability while the Gram-Schmidt process provides an occasion to perform successive orthogonalization [9].
In this paper the Gram-Schmidt orthonormalization algorithm and Givens Rotation algorithm together are used for the decomposition of matrix (A). Where the Gram Schmidt algorithm is used to calculate an orthogonal matrix (Q) and the Givens Rotation algorithm is used to calculate an upper triangular matrix (R). In the following subsections an explanation for Gram Schmidt and Givens Rotation algorithms, based on the steps of Gram Schmidt algorithm, the elements of the Q matrix (e1, e2, e3 and e4) are computed firstly, then the R matrix is computed by using the elements (ei) of matrix Q. While the R matrix is calculated at first by using the Givens Rotation algorithm, then the Q matrix is calculated depending on the elements computed in matrix R. Fig. 1 shows the outline for these algorithms.

2.1. Gram schmidt algorithm

A matrix A can be decomposed into the product of a Q orthogonal matrix and an R upper triangular matrix, where: A = QR, Fig. 2 illustrates the part of Q matrix computation from GS algorithm [10].

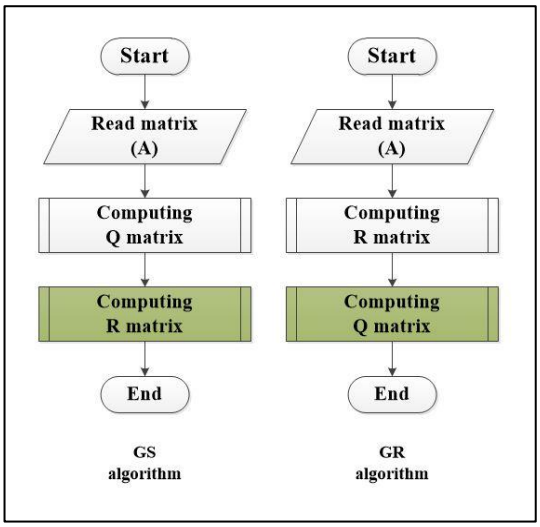

Fig. 1: Outline of GS and GR Algorithms.

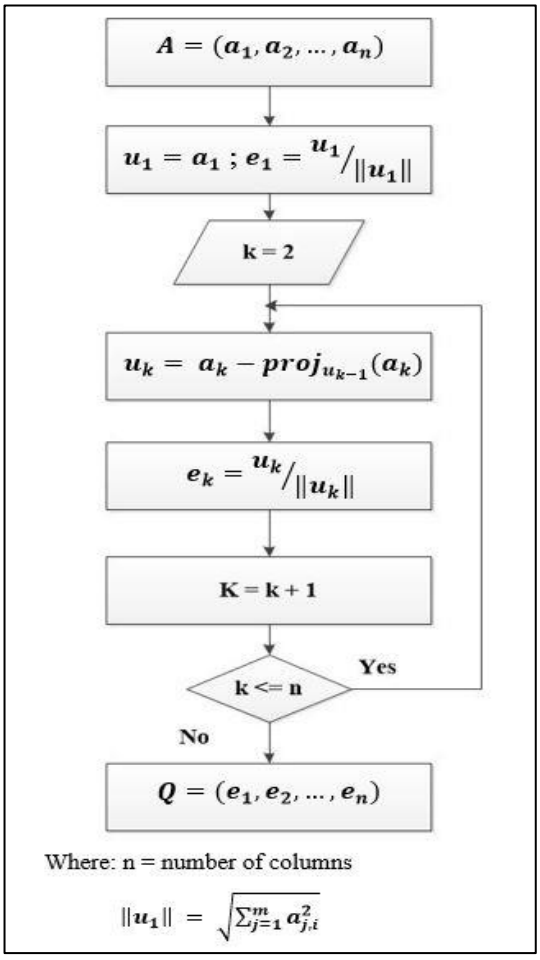

Fig. 2: Q Matrix Computation by GS Algorithm.

2.2. Givens rotation algorithm

The QR decomposition can be computed using Givens Rrotation [11]. Givens Rotation algorithm eliminates one element in the matrix at a time.

If matrix A consists of (4 × 3) elements,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

to compute the QR decomposition by Givens Rotation, the first step, is to use $a_{3,1}$ to eliminate $a_{4,1}$ by using equation (1) then, computing Cosθ and Sinθ using equations (2) and (3) respectively.

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

$$\text{Cos}\theta = \frac{a_{i-1k}}{r_{i-1,i}} \quad (2)$$

$$\text{Sin}\theta = \frac{a_{ik}}{r_{i-1,i}} \quad (3)$$

Where, a_{i-1k}, a_{ik} are elements of matrix A, k denotes to the column number, while, i denotes to the row number that is required to eliminate its element.

Step two, generating $G_{i-1,i}^{(k)}$ matrix (Givens Rotation matrix) using I matrix by adding values of Cosine and Sine into I matrix. Where $G_{33}^1 = \text{Cos}\theta, G_{34}^1 = \text{Sin}\theta, G_{43}^1 = -\text{Sin}\theta$ and $G_{44}^1 = \text{Cos}\theta$. As shown in equation (4).

$$G_{3,4}^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta_{3,4} & \sin\theta_{3,4} \\ 0 & 0 & -\sin\theta_{3,4} & \cos\theta_{3,4} \end{bmatrix} \quad (4)$$

Step three, computing first rotation using equation (5), by multiplying G^1 and A^0, to produce A^1.

$$A^i = G^i * A^{i-1} \quad (5)$$

Then repeating this process five times to compute R matrix. Equation (6) shows this process, and Fig. 3 shows the givens rotation algorithm [11].

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ X & \times & \times \\ X & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ X & \times & \times \\ X & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} X & \times & \times \\ X & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(1,2)}$$

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & X & \times \\ 0 & X & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ 0 & X & \times \\ 0 & X & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & X \\ 0 & 0 & X \end{bmatrix} \xrightarrow{(3,4)} R \quad (6)$$

```
For j = 1:n
  For i = m − 1:j + 1
      [sin, cos] = givens (A (i − 1, j), A (i, j))
    A (i − 1: i, j: n) = [cos  sin; −sin  cos]^T * A (i − 1: i, j: n)
  End
End
```

Fig. 3: Givens Rotation Algorithm.

After finding R matrix, equation (7) is used to compute Q matrix (orthogonal matrix) from G matrices.

$$Q = (G^i G^{i+1} G^{i+2} \dots \dots G^m)^{-1} \quad (7)$$

3. Multi core processor design

Specific processor design is proposed to compute the QR decomposition using Gram Schmidt algorithm and Givens Rotation algorithm. This processor design is based on multi core processor architecture, where it consists of two cores and each core is responsible on specific function as shown in Fig. 4. The first core processor is used to compute the Q matrix using a Gram Schmidt algorithm and, the second core processor is used to compute the R matrix using Givens Rotation algorithm to achieve high levels of parallelism that leads to higher throughput.

In this paper by using a multicore processor, the first core (Q core) is used to compute the Q matrix using Gram Schmidt algorithm, while the second core (R core) is used to compute the R matrix using the Givens Rotation algorithm. So that, both cores were working simultaneously to compute Q and R matrices, which gives a high level of parallelism. The details of each core processor is explained in the following subsections.

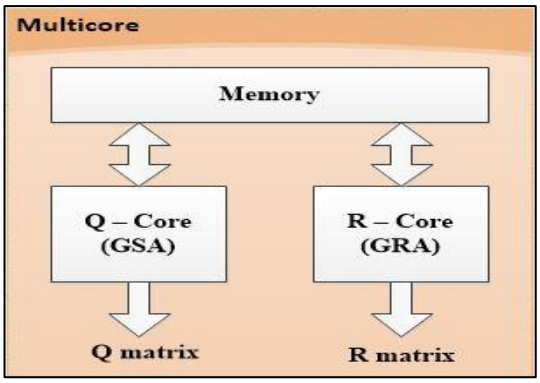

Fig. 4: Outline of Multicore Processor.

3.1. Design the Q core processor

This core processor is designed to perform QR decomposition using Gram Schmidt algorithm to compute the Q matrix only. A 32-bit architecture is used in this design for high accuracy and to decreases the percent of error in arithmetic operations where the fixed point method is used as system of numbers. As shown in Fig. 5, this design consists of several units, which they are the Register set, instruction pointer, instruction memory, data memory, control unit, stage 1, stage 2 and E-reg unit. Following a discussion and details for each of these units.

Register Set, is an internal memory, which consists of a set of storage locations, where each location consists of 32 bits (32 flip flops), the design consists of 32 registers. Instruction Pointer unit (IP), is a 32-bit register used to store 32-bits address of memory location from which an instruction to be fetched. In single cycle processor the memory divided into two parts instruction and data, the first part to store the instructions and the second part to store the data, this arrangement enables programmer to read instruction and data at the same time or in the same cycle, this type of design is known as Harvard memory architecture [12]. In this processor the memory is designed as four way enter leaving and the size of instruction memory is 1K byte while the size of data memory is 256 bytes. Control unit is responsible for controlling the data flow and an interface between the units. The proposed control unit is the largest unit. The control unit is designed to produce six control signals (regwrite, qen, lq1, lq2, lq3 and eout) details of these signals are shown in table 1.

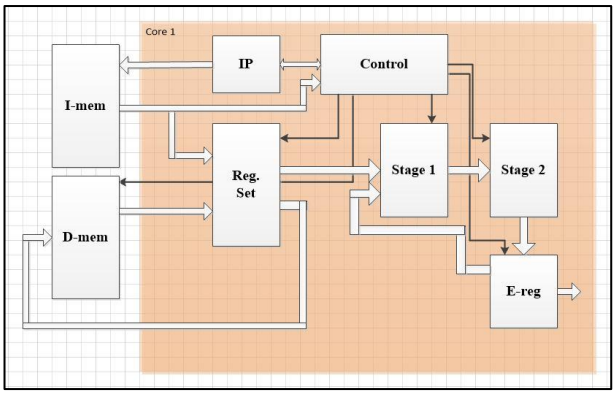

Fig. 5: Block Diagram of the Designed Processor.

Table 1: Control Signals

Sort	Signal Name	Description	Bits
1	regwrite	Writing into register (active high).	1
2	qen	Write into E unit	1
3	lq1	Enable latch one	1
4	lq2	Enable latch two	1
5	lq3	Enable latch three	1
6	eout	Enable output port	1

The main units of this design is stage 1 unit and stage 2 unit. Where, these two units are used to compute Q matrices using Gram Schmidt algorithm, u1 is computed by using stage 1, while, e1 is computed by using stage 2. Following details for each unit.
As shown in Fig. 6, stage 1 unit is consist of three parts to compute:

$$u_n = a_n - proj_{u_{n-1}}(a_n) \tag{8}$$

Where

$$proj_{u_{n-1}}(a_n) = (a_n * e_{n-1})\, e_{n-1} \tag{9}$$

This processor is designed for matrix A (4 x 4) elements, so, equation (8) can be rewritten as in equation (10) to become suitable for each column.

$$u_n = a_n - (a_n * e_1)\, e_1 - (a_n * e_2)\, e_2 - (a_n * e_3)\, e_3 \tag{10}$$

Where the initial values for each e_n is equal to zero. The stage 1 unit is designed depending on equation (8). As shown in Fig. 4 each part refers to "proj", so, the parts of stage 1 unit consist of eight multipliers and three adders connected in parallel, then, the subtraction operation is performed between $\boldsymbol{a_n}$ and (part 1, part 2 and part 3). From this unit u vector (4 x 1) will be produced and then sends it to the next stage unit under control signals, Fig. 7 shows the block diagram of stage 2.

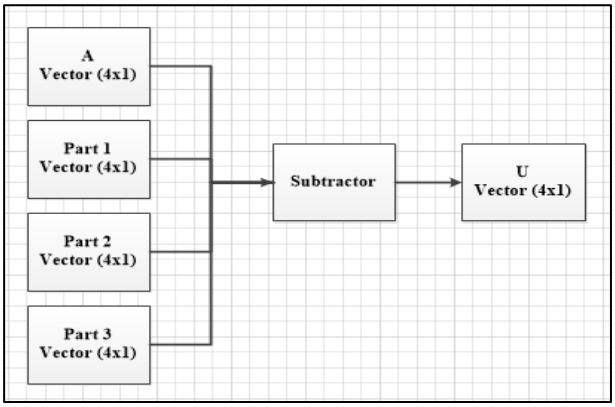

Fig. 6: Block -Diagram of Stage 1 Unit.

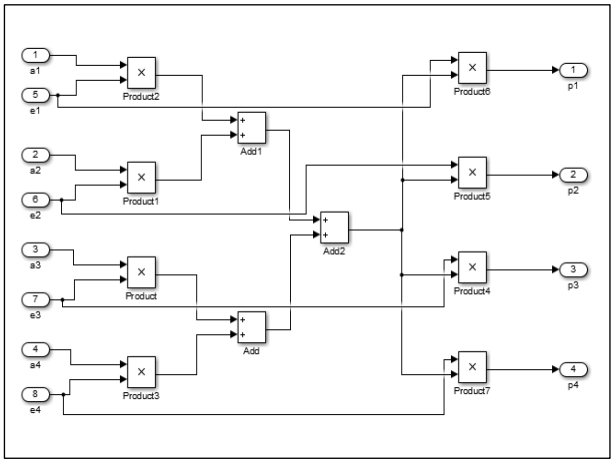

Fig. 7: Block Diagram of Stage 2 Unit.

Stage 2 unit is responsible for computing:

$$e_n = \frac{u_n}{norm\,(u_n)} \tag{11}$$

$$\text{Where } norm\,(u_n) = \sqrt{\sum u^2} \tag{12}$$

Fig. 8 shows the internal architecture for stage 2 unit. As shown in equations (11) and (12) the square root and division operations are needed to perform these equations. Coordinate Rotation Digital Computer algorithm (CORDIC) is used to calculate the square root operation. This algorithm depends on shift-add operations. To increase the speed of execution, multiplier is used. Fig. 9 shows an example for this method to compute a square root for number (x) which consists of 16-bits. 16 iterations are needed to compute the square root for number consists of 32-bits. So, to decrease the latency this stage is designed to work in parallel. Fig. 10 shows the block diagram for this design.

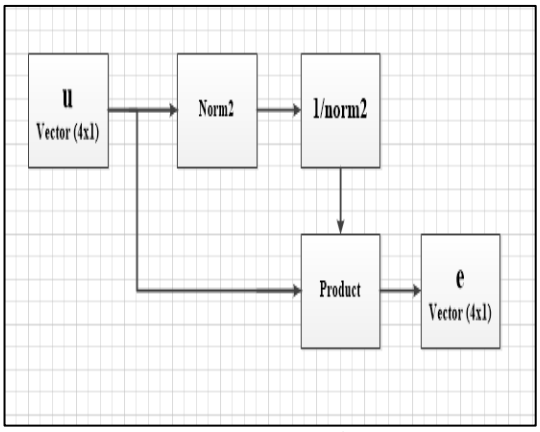

Fig. 8: Block -Diagram of Stage 2.

L	2^L	y	x= 12056	
		0	initial value	
7	128	0	128 x 128 > 12056	do nothing
6	64	64	64 x 64 < 12056	add 64 to $y_{initial}$ --> 64
5	32	96	$(64 + 32)^2$ < 12056	add 32 to last y --> 96
4	16	96	$(96 + 16)^2$ > 12056	do nothing
3	8	104	$(96 + 8)^2$ < 12056	add 8 to last y --> 104
2	4	108	$(104 + 4)^2$ < 12056	add 4 to last y --> 108
1	2	108	$(108 + 2)^2$ > 12056	do nothing
0	1	109	$(108 + 1)^2$ < 12056	add 1 to last y --> 109
-1	0.5	a.s.o.	and so on	and so on

Fig. 9: Example about CORDIC Algorithm.

Where: X equal to 12056 and the square root is 109.
L equal to N-bits / 2.
Y is square root value.

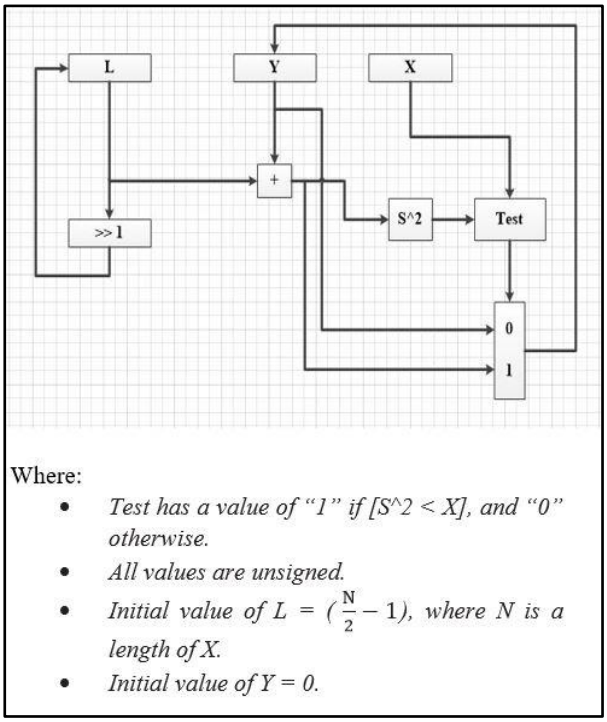

Fig. 10: Block Diagram for CORDIC.

The DSP divider is used to perform a division operation. This technique is used to achieve low latency. This type of divider supports only in top series of FPGA boards as Virtex-7 and Spartan-6 [13].
E-reg unit is a set of registers (16 registers) each one consists of 32-bits used to store the orthogonal matrix Q (Q = e). This unit directly connected with stage 2 to reduce the time for data transfer between them.

3.2. Design the R core processor

The R core processor is designed based on Givens Rotation algorithm to compute the R matrix. The design of the R core processor is consisting of four stages and each stage consists of cells, where these cells are responsible on rotation of columns.
The classic implementation of Givens Rotation algorithm is shown in Fig. 11, beginning with zeroing the bottom most element of first column and continues up in the same column, then repeating this procedure on next column and so on to complete the R matrix. But in this paper, the higher level of parallelism is used to rotate many rows in the same time as shown in Fig. 11 [4].

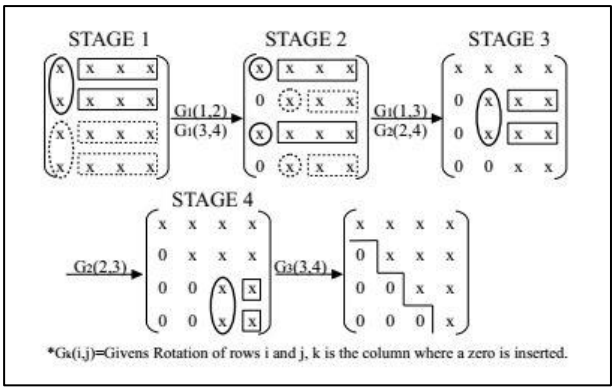

Fig. 11: Higher Level of Parallelism for GR.

It is clear from Fig. 11; four stages are required to compute R matrix for matrix A (4 × 4). Stages one and two are consisting of two cells while stages three and four are consisting of one cell only as shows in Fig. 12.

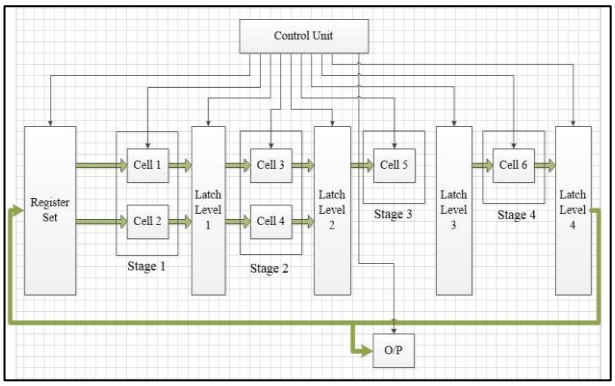

Fig. 12: Block Diagram of R Core Processor.

Each cell consists of two parts, the function of the first part is to calculate the values of Sine and Cosine based on equations (2 and 3), as shows in Fig. 13. It is clear from Fig. 13, that the calculation of Sine and Cosine require math operations, which they are multiplication, division and square root. The CORDIC algorithm is used in calculating the square root in the design of this part. While the function of the second part is to perform the matrix multiplication using several of DSP multipliers and adders. The cells are needed to 8 clock cycles to perform its function.

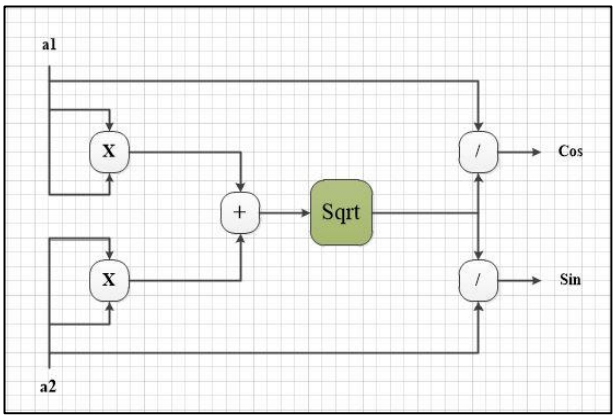

Fig. 13: Internal Architecture of Cell.

4. Design results

The proposed design of multi core 32-bits processor has been written using Verilog Hardware Description Language and implemented using Virtex- 7 FPGA board. 32-bits fixedpoint number representation is used in this design. A testbench is created for testing the results of QRD with matlab program. The Isim behavior simulation result shows that the total latency are 4 clock cycles. Matrix A shown below is given from matlab program as test matrix for this design:

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

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

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

And

$$R = \begin{bmatrix} 1.7321 & 1.1547 & 1.1547 & 1.1547 \\ 0 & 1.2910 & 0.5164 & 0.5164 \\ 0 & 0 & 1.1832 & 0.3381 \\ 0 & 0 & 0 & 1.1339 \end{bmatrix}$$

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

$$Q = \begin{bmatrix} 0.5774 & 0.2582 & 0.1690 & -0.7560 \\ 0.5774 & 0.2582 & -0.6762 & 0.3779 \\ 0.5774 & -0.5166 & 0.5071 & 0.3780 \\ 0 & 0.7747 & 0.5071 & 0.3780 \end{bmatrix}$$

And

$$R = \begin{bmatrix} 1.7322 & 1.1548 & 1.1548 & 1.1548 \\ 0 & 1.2911 & 0.5163 & 0.5163 \\ 0 & 0 & 1.1832 & 0.3380 \\ 0 & 0 & 0 & 1.1339 \end{bmatrix}$$

Because this design of multi core processor is pipelining architecture, it is possible to produce orthogonal and upper triangular matrices every 4 clock cycles only.

Table 2 shows a comparison between this work and with previous works. This comparison based on the maximum run frequency, data word length, type of algorithm (Gram Schmidt, Householder Transformation and Givens Rotation) and latency. It is clear from this table that this design has the lower latency from other works and has a higher throughput. These results an achieved because the high parallelism of the designed work.

Fig. 14, shows the screen shot of RTL (Register Transfer Level) for this design by ISE Design Suite 14.7 program, while Fig. 15 shows the capture of test bench of the result.

Table 2: General Comparison

Items	[6]	[8]	This Work
algorithm	MGS	HT	GS & GR
Word length	14 bits	-	32 bits
Latency (clk)	35	8	4
Throughput (QRD/s)	11.42 M	22 M	50 M
Max frequency	-	110 MHz	200 MHz

Where: MGS denotes Modified Gram Schmidt algorithm, HT denotes Householder Transformation algorithm, GR, GS denotes Givens Rotation and Gram Schmidt algorithms.

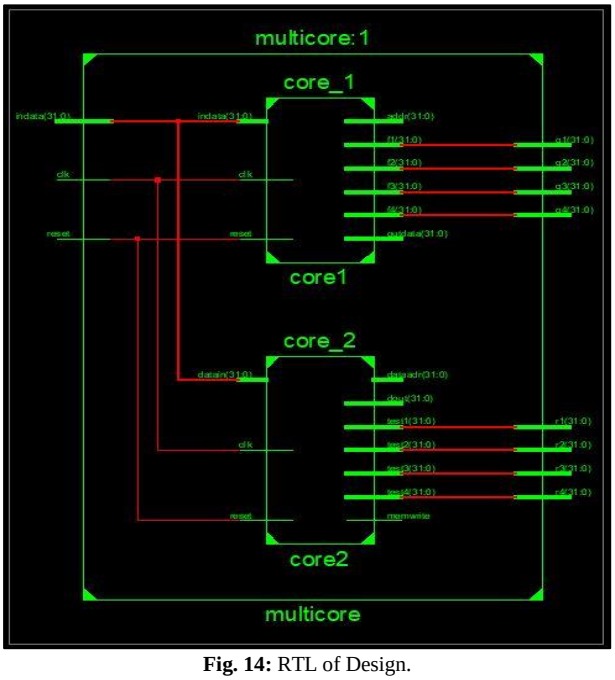

Fig. 14: RTL of Design.

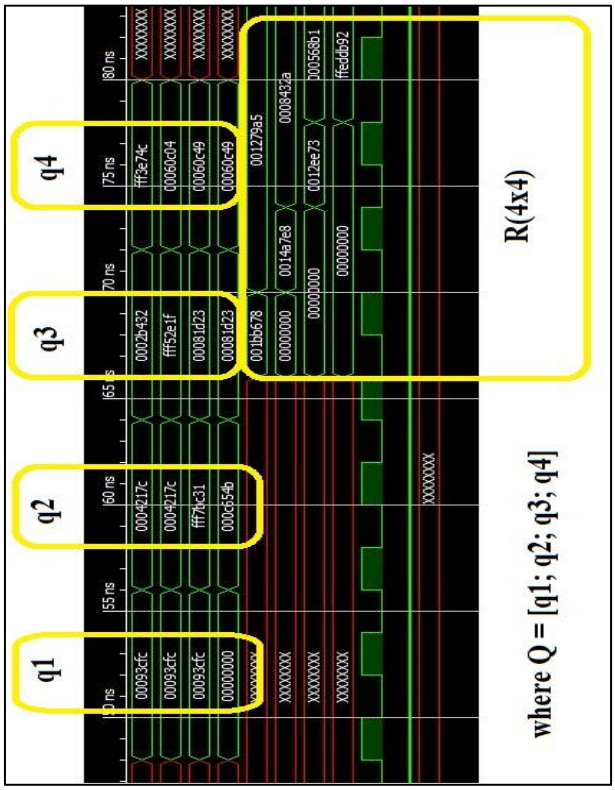

Fig. 15: Capture of Result.

5. Conclusions

In this work 32-bits multi core processor is designed to perform QR decomposition based on two algorithms which they are Gram Schmidt and Givens Rotation, where each one of them is used to implement by one core. The first core is working using Gram Schmidt and it is responsible to calculate Q matrices, while the second core is responsible to calculate R matrices. The two cores working simultaneously, so that in the same time the Q matrix and the R matrix are calculated. This design of parallelism in computing the Q and R matrices simultaneously achieves a low latency and high throughput.
This design is implemented using Virtex-7 development board and Verilog HDL. By this multicore processor the throughput is increased to 50 M matrix per second at run frequency 200MHz.

References

[1] K. Sarrigeorgidis and J. Rabaey, "A scalable configurable architecture for advanced wireless communication algorithms," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 45, no. 3, p. 127–151, 2006. https://doi.org/10.1007/s11265-006-9762-9.

[2] S. Chan and X. Yang, "Improved approximate QR-LS algorithms for adaptive filtering," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 51, no. 1, pp. 29 - 39, 2004. https://doi.org/10.1109/TCSII.2003.821514.

[3] Z.-Y. Huang and P.-Y. Tsai, "Efficient Implementation of QR Decomposition for Gigabit MIMO-OFDM Systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 10, pp. 2531 - 2542, 2011. https://doi.org/10.1109/TCSI.2011.21237700.

[4] S. D. Munoz and J. Hormigo, "High-throughput FPGA implementation of QR decomposition," *IEEE Transactions on Circuits and Systems II*, vol. 6, no. 9, p. 1, 2015. https://doi.org/10.1109/TCSII.2015.2435753.

[5] 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. https://doi.org/10.1109/APCCAS.2008.4746151.

[6] R. C. H. Chang and C. H. Lin and K. H. Lin and C. L. Huang and F. C. Chen, "Iterative QR decomposition architecture using the modified gram–schmidt algorithm for MIMO systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 5, pp. 1095-1102, 2010. https://doi.org/10.1109/TCSI.2010.2047744.

[7] Dongyeob Shin, Ji-Hwan Yoon, "Gram-schmidt tailed high-throughput QR decomposition architecture for MIMO detector," in *International SoC Design Conference (ISOCC)*, Jeju, South Korea , 2014.

[8] R. Gangarajaiah, O. E. Liu and Liang, "An adaptive QR decomposition processor for carrier-aggregated LTE-A in 28-nm FD-SOI," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 7, pp. 1914 - 1926, 2017. https://doi.org/10.1109/TCSI.2017.2658729.

[9] S. S. Omran and A. K. Abdul-abbas, "Design of 32-bits RISC processor for hardware efficient QR decomposition," in 2018 International *IEEE Conference on Advance of Sustainable Engineering and its Application (ICASEA)*, Wasit - Kut, Iraq, Iraq, 2018. https://doi.org/10.1109/ICASEA.2018.8370958.

[10] M. Parker, V. Mauer and D. Pritsker, "QR Decomposition using FPGAs," in *EEE National Aerospace and Electronics Conference and Ohio Innovation Summit*, Dayton, OH, USA, 2016. https://doi.org/10.1109/NAECON.2016.7856841.

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

[12] S. S. Omran and H. S. Mahmood, "Pipelined MIPS processor with cache controller using VHDL implementation for educational purposes," in *IEEE International Conference on Electrical Communication, Computer, Power, and Control Engineering*, Mosul, Iraq, 2014.

[13] xilinx, "www.xilinx.com," Xilinx Inc., 2017. [Online]. Available: https://www.xilinx.com/products/intellectual-property/divider.html#overview.