Review on architectures and applications of cores based on CORDIC algorithm

Priyanka Shakya[1] and RCS Chauhan[2]
M. Tech Scholar[1] Associate Professor[2]
Institute of Engineering and Technology, Lucknow UP 226021

Abstract: *The advancement in technology helps in the business and organizations by saving time and aiding with minimal hardware equipment with the fast response. Similarly advancement in the CORDIC technology has helped a lot in decreasing the hardware utilisation and increase in the throughput. CORDIC (Coordinate Rotation Digital Computer), Jack E. Volder's innovation led to its efficient application in the field of communication, digital signal processing and robotics and as well as in image processing. CORDIC architecture beauty lies in the use of simple shift and adds operations for complex computations such as trigonometric operations, exponential, logarithmic, square root, division and evaluation of eigenvalues. Subsequently, the growth of the CORDIC architecture has led to its expansion towards diverse application from simple to complex scientific applications. In this paper work, we discussed the upgradation of the CORDIC architecture and its features along with the utilisation in various areas.*

Keywords: Cordic, framework, trigonometric, exponential, logarithmic

Introduction

Coordinate Rotation Digital Computer is abbreviated as CORDIC. The postulations about the concept of CORDIC computations are based on the simple and efficient theory of two-dimensional configurations. Jack E. Volder led to the pioneering of the CORDIC framework in history in 1959 [1]. His discovery provided an iterative approach for the implementation of various trigonometric functions, logarithmic functions, multiplications and many other tasks. CORDIC has become the useful architecture and has made a significant progress in the designing of algorithms and hardware structures that provide high throughput and less hardware complexity. Implementation of square root and exponential functions can also realized through slight modification in the algorithm suggested by Volder [1] and these CORDIC alterations were given by Walther in 1971[2]. Cochran [3] marked the usage of CORDIC architecture in scientific calculator and revealed that CORDIC is the better choice.

The rest of the paper is divided into different sections as follows. The Section II consists of the basic CORDIC architecture and its operation, representing the key features and functions related to coordinate transformation based on rotation mode and vectoring mode. In Section III, we consider the important evolutions of CORDIC algorithms and hardware structures, covering the higher-radix CORDIC, angle recoding methods, hybrid and pipeline implementation. Section IV includes the important applications of CORDIC framework in the field of signal processing, communications, robotics and graphics. The upcoming future and conclusion are included in Section V.

The CORDIC Algorithm

The two dimensional mathematical vector $p_0 = [x_0 y_0]$ can be turned through an angle θ to obtain the rotated vector $p_n = [x_n y_n]$. This can be implemented by the method of matrix multiplication $p_n = R p_0$, where R is the rotation matrix.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad (1)$$

By calculating out the cosine term in (1), the resultant matrix can be modified as

$$R = \left[(1+\tan^2\theta)^{-\frac{1}{2}}\right]\begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \quad (2)$$

and can be defined as a product of a scaling factor $d = [(1+\tan^2\theta)^{-\frac{1}{2}}]$ with pseudo rotation matrix R_c given by

$$R_c = \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \quad (3)$$

When the vector p_0 is rotated by an angle θ and the pseudo rotation operation changes its magnitude by a factor $d = \cos\theta$, to produce a pseudo-rotated vector $p'_n = R_c p_0$.

To accomplish the task of minimum hardware requirement for the rotation, the core conclusions used in CORDIC arithmetic are to

1) break down the rotations into series of basic rotations according to the given angles that can be designed using less number of hardware components.
2) remove scaling which involves mathematical operations such as division and square root.

The second method is useful because the factor does not provide information about the rotation angle and it only gives information about the magnitude.

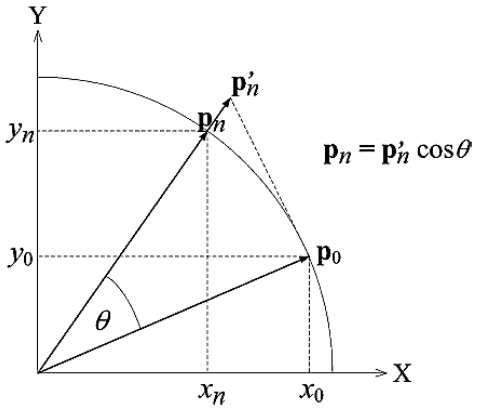

Figure 1. Vector rotation

Decomposition of rotation angle- The CORDIC algorithm rotates the given angle in the sequence of small rotations. It iteratively breaks down the rotation angle into a set of small already defined angles $\alpha_i = arc\tan(2^i)$ such that $\tan\ \alpha_i = 2^i$ can be implemented by only shifting through i bit locations in hardware. CORDIC does not perform rotations directly instead it uses series of micro-rotations through the angle α_i, where

$$\theta = \sum_{i=0}^{n-1} \sigma_i \alpha_i \text{ and } \sigma_i = \pm i \tag{4}$$

This satisfies the convergence theorem of CORDIC algorithm [3]: $\alpha_i - \sum_{j=i+1}^{n-1} \alpha_j < \alpha_{n-1}, \forall i, i = 0,1,2,\ldots,n-2$. But, this theorem can only be used for $-1.7329 \le \theta \le 1.7329$ range. Since $\sum_{i=0}^{\infty} \alpha_i = 1.74328$. Therefore, the angles in the first and fourth quadrants fall in the range of this angular decomposition (4).

One can use the non-restoring decomposition [6]-

$$w_0 = 0 \text{ and } w_{i+1} = w_i - \sigma_i \alpha_i \tag{5}$$
$$\sigma_i = 1 \text{ when } w_i \ge 0 \text{ and } \sigma_i = -1 \text{ otherwise.}$$

The rotation matrix according to the selected angle α_i for *i*th iteration is given by

$$R(i) = d_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \tag{6}$$

where $d_i = 1/\sqrt{(1+2^{-2i}}$ scaling factor and pseudo-rotation matrix is

$$R_c(i) = \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \tag{7}$$

Avoidance of scaling factor- When the scaling factor K is removed from the micro-rotations, it produces the rotated vector as $p'_n = R_c p_0$ instead of the actual angular vector $p_n = dR_c p_0$ where scaling factor is given by

$$d = \prod_{i=0}^{n} d_i = \prod_{i=0}^{n} 1/\sqrt{(1+2^{-2i})} \tag{8}$$

The scale factor *d* is not affected by the direction in which the micro-rotations take place and monotonously decreases, thus converging *d* to value ~1.646760. Thus the value of the resultant output can be scaled by multiplier factor *d* instead of scaling at each micro-rotation.

The basic CORDIC iterations are given as

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i 2^{-i} y_i \\ y_{i+1} &= y_i + \sigma_i 2^{-i} x_i \\ z_{i+1} &= z_i - \sigma_i \alpha_i \end{aligned} \tag{9}$$

The CORDIC computations are performed in two modes-

- rotation mode (RM)
- vectoring mode (VM)

These depend on the direction of micro-rotation. In the rotation mode, the given coordinate vector p_0 is rotated by an angle θ to obtain a new coordinate vector p'_n . The direction of each micro-rotation σ_i in this mode is decided by z_i and its sign: if sign of z_i is positive, then $\sigma_i = 1$ otherwise $\sigma_i = -1$. The vectoring mode is used when the original vector p_0 is rotated towards x axis until the y coordinate reaches to zero value. In this mode, the sign of y_i coordinate decides the direction in which the rotation should take place. If y_i is positive then then $\sigma_i = -1$ otherwise $\sigma_i = 1$. Figure 2 shows the hardware structure of CORDIC architecture for single iteration.

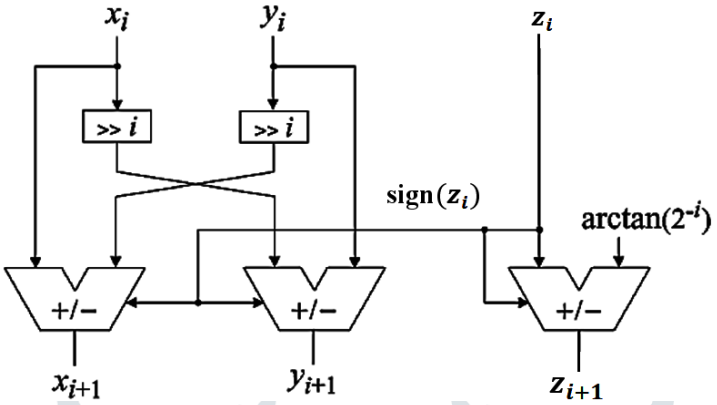

Figure 2. CORDIC structure for single iteration

Table I
Generalized CORDIC Algorithm

m	Rotation mode	Vectoring mode
0	$x_n = d(x_0 \cos z_0 - y_0 \sin z_0)$	$x_n = d\sqrt{x^2{}_0 + y^2{}_0}$
	$y_n = d(y_0 \cos z_0 + x_0 \sin z_0)$	$y_n = 0$
	$z_n = 0$	$z_n = z_0 + \tan^{-1} {}^{y_0}/_{x_0}$
1	$x_n = x_0$	$x_n = x_0$
	$y_n = y_0 + x_0 z_0$	$y_n = 0$
	$z_n = 0$	$z_n = z_0 + {}^{y_0}/_{x_0}$
-1	$x_n = d_h(x_0 \cosh z_0 - y_0 \sinh z_0)$	$x_n = d_h\sqrt{x^2{}_0 - y^2{}_0}$
	$y_n = d_h(y_0 \cosh z_0 + x_0 \sinh z_0)$	$y_n = 0$
	$z_n = 0$	$z_n = z_0 + \tanh^{-1} {}^{y_0}/_{x_0}$

Generalized form of the CORDIC architecture

Walther declared that CORDIC computations can be altered to evaluate hyperbolic functions [2] and rephrase the CORDIC equations in to a simplified and compact form. This technique is able to achieve the mathematical functions in circular, hyperbolic and linear coordinate systems. The generalized form of equations includes the variable m, which gives different values for various coordinate systems. The generalized CORDIC algorithm is gives as:

$$\begin{aligned} x_{i+1} &= x_i - m\sigma_i 2^{-i} y_i \\ y_{i+1} &= y_i + \sigma_i 2^{-i} x_i \\ z_{i+1} &= z_i - \sigma_i \alpha_i \end{aligned} \tag{10}$$

where

$$\sigma_i = \begin{cases} \text{sign of } z_i \text{ , for rotation mode} \\ -\text{sign of } y_i \text{, for vectoring mode} \end{cases}$$

$$\alpha_i = \tan^{-1} 2^{-i}$$

The value of m can be allotted as 1, 0 and -1 for circular, linear and hyperbolic systems.

ADVANCED CORDIC ALGORITHMS AND ARCHITECTURES

The computation in CORDIC algorithm is serially formulated because of two main constrictions: 1) the micro-rotation can be executed on the intermediate vector only after execution on the previous vector for any iteration 2) the starting of the $(j+1)th$ iteration can take place only after the execution of jth iteration, because the value of σ_{j+1} for $(j+1)th$ iteration can be decided after the jth iteration. In order to refine the second constriction several undertakings have been considered to enhance the values of σ_j in accordance to the small micro-rotation angles [4], [5]. Though the iterations in CORDIC cannot be operated in parallel mode due to the first constriction. Several attempts are tried to acquire partial parallelization [6] by conglomerating pair of conventional iterations into single unique composite iteration in CORDIC that gives enhanced area and less delay. But this merging affects the accuracy of the system such that the number of iterations cannot be extended and the error invoked becomes intolerable. The first constriction can be solved by direct expansion of micro-rotations in parallel framework of CORDIC iterations, but it will lead to increase in complex computations. It causes degradation of the simplicity of CORDIC algorithm [7], [8]. Thus the parallel realization of CORDIC is not accomplished fully and the various pipelined CORDIC structures have been proposed in order to enhance the computational speed [9].
The CORDIC algorithm has direct rate of convergence that is to get the n bit precision at the output, it requires (n+1) iterations. However total latency depends on the product of iteration period and word length. The operations of CORDIC have restricted speed due to the iterations count required or the clock period. In each micro-rotation the clock duration relies on the large propagation time for adding/subtracting. In order to reduce the large delay during iterations, fast adders are used with the trade-off in large silicon area. Latency and the large propagation delay can be minimized by using carry save architecture [10].
To resolve the constrictions related to latency, different methodologies have been adopted and described in the proposed paper works. Different types of CORDIC algorithms are categorized as, radix-4 CORDIC, the recoding of angle in CORDIC, redundant CORDIC and hybrid rotation methods which are considered in the following later sections.

Radix-4 based CORDIC algorithm

This algorithm is given as

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i 4^{-i} y_i \\ y_{i+1} &= y_i + \sigma_i 4^{-i} x_i \\ w_{i+1} &= w_i - \sigma_i \alpha_i \end{aligned} \quad (11)$$

where

$$\sigma_i = \{-2, -1, 0, 1, 2\}$$
$$\alpha_i = \tan^{-1} 4^{-i}$$

Scaling factor is $d_i = 1/\sqrt{(1+\sigma_i^2 4^{-2i})}$ (12)

CORDIC algorithm radix-4 requires $n/2$ number of iterations to get the n-bit precision at the output. It requires half of the micro-rotations required by the radix-2 CORDIC but the complexity of the computation time and components requires increases as the value of σ_i has to be selected from the five values. The scale factor depends on σ_i which changes the value of K according to the five values. Some of the proposed methods are implemented for the adjustment of value of K using simple shift and operations [11], [12]. Higher radix algorithms [13] have also been suggested as the solution for latency but it causes the increase in the size of the lookup table which stores the rotation angle and affects the scaling factor than the lower order radix algorithms.

Angle Recoding (AR) Methods

This method involves the breaking the primary rotation angle as the group of set of fundamental micro-rotation angles in CORDIC iterations. This method is suitable for various applications such as processing of image signal and digital signal processing where the rotation of angle is the main task performed, such as in calculation of discrete Fourier and cosine transforms, etc.

Angle Recoding: In the previous CORDIC algorithms, the primary angle is represented as the combination of the set of n rotation angles that belong to the set as $S = \{(d.\arctan(2^{-j}) : d \in \{-1,1\}, j \in \{1,2,\dots,n-1\}\}$ where main rotation angle is given as $\theta = \sum_{j=0}^{n-1}[d_j \tan^{-1}(2^j)]$. In angle recoding technique, this limitation is removed by introducing zero to the range of combination which helps in obtaining the rotation angle using less number of iterations for term $d.\arctan(2^{-j})$ using combination $d \in \{-1,0,1\}$. Then the angle set for angle recoding techniques is represented as $S_{AR} = \{\left(d.\arctan(2^{-j}) : d \in \{-1,0,1\}, j \in \{1,2,\dots,n-1\}\right\}$. Hu and Naganathan proposed the recoding of angle based on greedy algorithm [14] given in table II. The backward recoding technique has also been proposed for vectoring mode [15].

Table II
Angle Recoding Algorithm

Initialize: $\emptyset_0 = \emptyset, \mathrm{d_j} = 0$ for $0 \le \mathrm{j} \le (\mathrm{n}-1)$ and $\mathrm{k} = 0$.

Repeat until $|\emptyset_\mathrm{k}| < \tan^{-1}(2^{-\mathrm{n}+1})$ do:

1. Choose $\mathrm{j_k}$, $0 \le \mathrm{j_k} \le (\mathrm{n}-1)$ such that
$$\left||\emptyset_\mathrm{k}| - \tan^{-1}(2^{-\mathrm{j_k}})\right| = \min_{0\le \mathrm{j} \le (\mathrm{n}-1)} \left||\emptyset_\mathrm{k}| - \tan^{-1}(2^{-\mathrm{j}})\right|$$
2. $\emptyset_{\mathrm{k}+1} = \emptyset_\mathrm{k} - \mathrm{d_{j_k}} \tan^{-1}(2^{-\mathrm{j_k}})$, where $\mathrm{d_{j_k}} = \mathrm{sign}(\emptyset_\mathrm{k})$

Extended Angle Recoding: This method represents the angle set as $S_{EAR} = \{\left(\arctan(d_1 2^{-j_1} + d_2 2^{-j_2}) : d_1, d_2 \in \{-1,0,1\}\ and\ j_1, j_2 \in \{1,2,\dots,n-1\}\right\}$. C.S. Wu proposed the technique of extended angle recoding which provides better efficiency and error free output. The coordinates for this angle recoding are given as

$$x_{i+1} = x_i - [d_1(i)2^{-j_1(i)} + \sigma_2(i)2^{-j_2(i)}]y_i$$
$$y_{i+1} = y_i + [d_1(i)2^{-j_1(i)} + \sigma_2(i)2^{-j_2(i)}]x_i \quad (13)$$

These are the pseudo-rotated vectors and these are to be scaled by the factor $K_i = [1 + (d_1(i)2^{-j_1(i)} + d_2(i)2^{-d_2(i)})^2]^{-1/2}$ to produce the rotated vector. The recoding of angle consists of the modification of micro-rotation angle and the scaling factor and these both can be implemented in the same framework as given in [16] and represented by following figure.

$$\tilde{x}_{i+1} = \tilde{x}_i - [k_1(i)2^{-s_1(i)} + k_2(i)2^{-s_2(i)}]\tilde{y}_i$$
$$\tilde{y}_{i+1} = \tilde{y}_i + [k_1(i)2^{-s_1(i)} + k_2(i)2^{-s_2(i)}]\tilde{x}_i \quad (14)$$

where $\tilde{x}_0 = x_{R_m}$ and $\tilde{y}_0 = y_{R_m}$, $k_1, k_2 \in \{-1,0,1\}$ and $\mathrm{q_1}, \mathrm{q_2} \in \{1,2,\dots,\mathrm{n}-1\}, \mathrm{R_m}$ represents required number of micro − rotations.

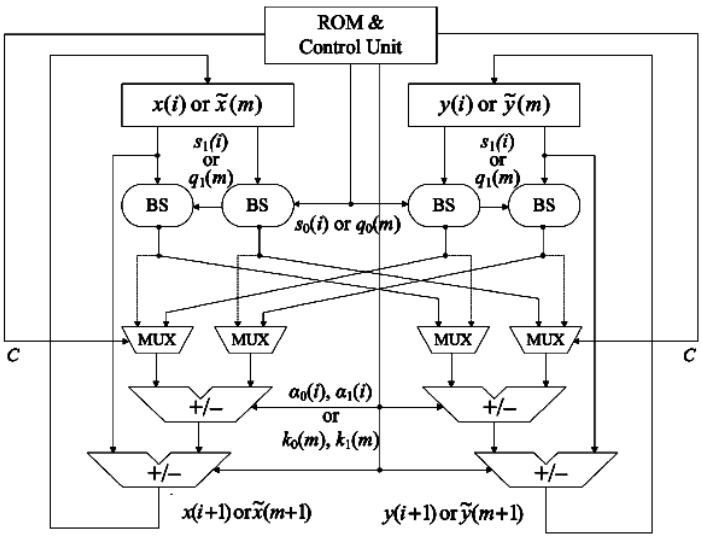

Figure 3. Extended angle recoding based CORDIC framework. BS is barrel shifter and C indicates control signals.

Hybrid CORDIC: According to the conventional CORDIC method, rotation angle can be represented by the combination of the sequence of micro-rotation angles given by the arctangents. However, radix-2 decomposition evaluates rotation angle as the combination of angles of the set $\{2^{-j}, j \in \{1,2,\dots,n-1\}\}$ where rotation angle is $\sum_{j=0}^{n-1}[b_j 2^j]$, $b_j \in \{0,1\}$. But in conventional CORDIC this technique is not used, instead arctangents are used as they minimize hardware complexity by using only shift and add operations. In the hybrid CORDIC, the basic idea is that the core angle is broken down as coarse and fine angles such that for fine values $\alpha_j = \tan^{-1} 2^{-j}$ when $j > [n/3 - 1]$, and for coarse angle $\alpha_j = 2^{-j}$, since $\tan^{-1} 2^{-j} \approx 2^{-j}$ for large values of j.

In angular decomposition, the angular set is represented as the combination of primary and secondary angles. The set is represented by $S = S_1 \cup S_2$. $S_1 \in= \{(\arctan(2^{-j}), j \in \{1,2,\dots,t-1\}$ and $S_2 \in= \{(2^{-k}, k \in \{t, t+1 \dots, n-1\}$ and value of k is very large such that $\tan^{-1} 2^{-k} \approx 2^{-k}$. Total rotation angle is given as

$$\theta = \theta_M + \theta_L \quad (15)$$

where θ_H and θ_L are coarse and fine angles.

$$\theta_H = \sum_{i=1}^{p-1} \sigma_i arctan(2^{-i}), \sigma_i \in \{-1,1\}$$

$$\theta_L = \sum_{i=p}^{n-1} d_i 2^{-i}, d_i \in \{0,1\}$$

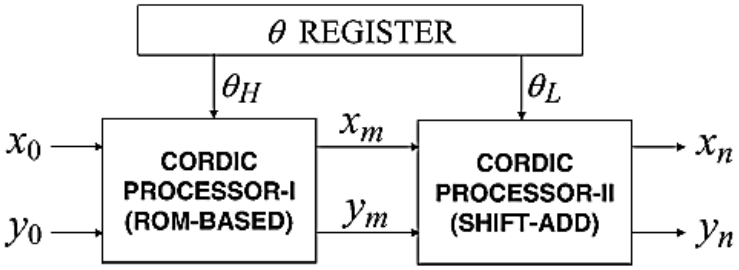

Figure 4.Architecture for as Hybrid CORDIC framework [5]

The framework of hybrid CORDIC representing coarse and fine angles is represented in the figure 4. It includes two processors I and II, where processor I evaluates the ROM based look up table with addition operation in order to find angle of rotation and processor II evaluates the fine sub angles followed by shift and add operations. Formulation of fine rotation angles does not involve the depiction of direction of micro-rotation thus reducing the hardware components used. The ROM based look up table for coarse phase implementation reduces the latency of the hybrid CORDIC. Some other techniques that implement the designing of two processors is suggested in [17] which provides high precision rotation angle and ROM size is about $n2^{n/5}$ bits.

The coarse and fine sub angles can also be evaluated using only shift and add operations without the use of look up tables [18,19]. In suggested paper work processor I performs the iterations similar to conventional CORDIC for one-third iterations and remaining process is performed by processor II. To reduce latency, the coarse-fine phase calculation have been use to generate sine and cosine angles [18]-[22] and also for rectangular to polar conversion [23], [24].

Pipelined Architecture of CORDIC: For high-throughput evaluation of sine and cosine wave generation, pipelined structure is much suitable and it is also used for designing filters; for simple computation of discrete transforms and other digital signal processing applications. The pipelined CORDIC structure is shown in the figure 5. It consists of several stages of conventional CORDIC as the basic unit.

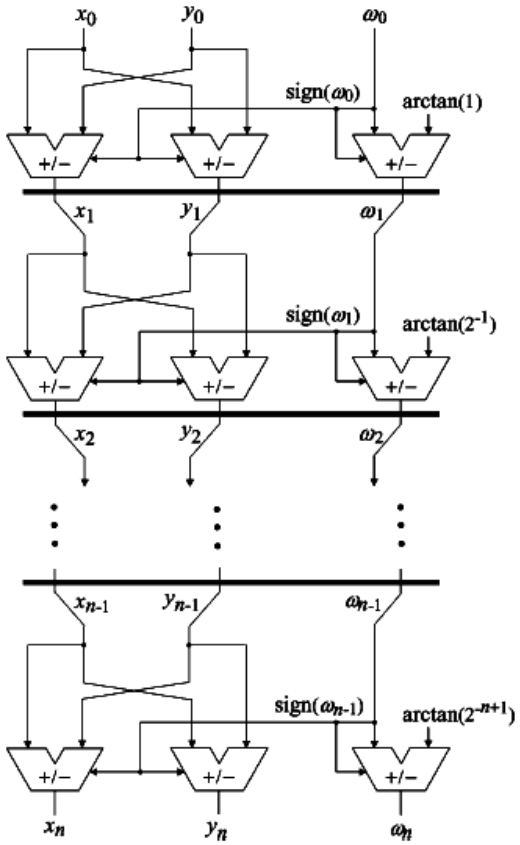

Figure 5. Pipelined architecture of CORDIC

The parallel architecture consists of the shifters that are fixed at each stage and these can be reduced in parallel structure by hardwiring them with the adders. Thus the path delay of the pipelined structure depends on the add/subtract operations of each stage. If three adders are used in figure 7 then total delay depends on $T_{ADD} + T_{MUX} + T_{2C}$ where T_{ADD} is addition time, $T_{MUX}$is time for multiplexing and T_{2C} is time for 2's complement calculation. The multiplexing task can be eliminated as the sign of micro-rotation is predetermined for the known angles, thereby minimizing path delay. Thus the path delay totally depends on the time consumed by the addition operation. Pipelined CORDIC structures are used for filter designing; discrete transforms computations and various applications of signal processing for high throughput [26]–[29].

APPLICATIONS OF CORDIC

CORDIC architecture provides us with an ease to evaluate the variety of parameters in signal processing, communication, robotics and graphics. It provides framework that is used for rotation of a coordinates in circular, hyperbolic or linear coordinate systems. It is also used for generation of sine and cosine waveforms, arithmetic operations such as multiplication and division, and rotation of angle, geometric calculations, logarithms, exponentials and square root [30], [31], [32].

Signal Processing and Image Processing Applications: CORDIC architecture has been used in various digital signal processing techniques such as sine and cosine wave generation, calculation of discrete transforms such as Hartley transforms [33],[34], [35], cosine transforms [36]-[40], Fourier transforms [41],[42], [43], [44], sine transform [45]-[48], and chirp transform [49]. CORDIC architecture has also played an important role in image processing schemes such as for spatial domain image rotation, Hough transform, contrast adjustment and logarithmic evaluation for line exposure [50], [51]. There are also some other suggested works in [52], [53], [54], [55].

Applications in Communication: CORDIC architecture is used for the construction of hardware module of communication system which can be used for different applications such as for generating the mixed signals performed by rotation mode of CORDIC and evaluating the angle and frequency parameters performed by vectoring mode of CORDIC. Various communication applications are described as follows.

Analog and Digital Modulation: Digital modulation is performed by using rotation mode in CORDIC as depicted in the figure 6.

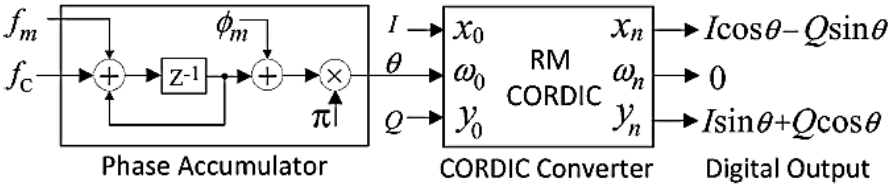

Figure 6. A general structure of digital modulation based on CORDIC. I is in-phase signal and Q indicate quadrature signal.

It generates the phase based on expression $[(\sum(f_c + f_m) + \emptyset_m).\pi]$ for f_c is the carrier frequency, f_m is the message frequency and $\emptyset_m$ is section of modulation phase. In order to design efficient modulation system, parameters such as f_c, f_m and $\emptyset_m$ must be carefully selected. Thus the system designed can be used for analog amplitude modulation, phase modulation and frequency modulation and digital modulations include amplitude shift keying, phase-shift keying, and frequency- shift keying modulators.

Digital CORDIC Synthesizer: This synthesizer is used to produce the sinusoidal waves straightway in digital domain.

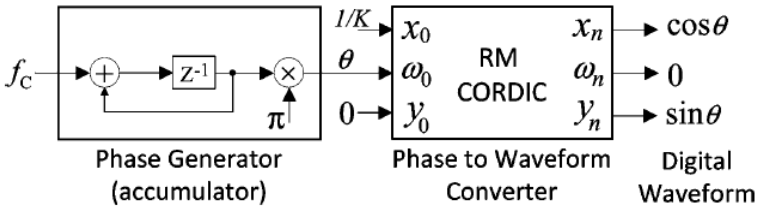

Figure 7. CORDIC implementation of digital synthesizer.

The digital synthesizer implemented using CORDIC framework has been depicted in the figure 7. The framework of DDS consists of phase generator whose output is given to phase to waveform converter [56]. The phase generator produces phase in accordance with the expression $(\sum f_c).\pi$ where f_c is the carrier frequency and then generated phase is sent to waveform converter. This synthesizer is designed using rotation mode CORDIC [57], [58], [59], [60].

Implementation of CORDIC in the field of Robotics and Graphics: CORDIC has formed an integral part in the area of robotics for rotation purpose as well as for detection of obstacles.

Basic kinematics: Robot consists of various joints and links either for movement or prismatic motion. In order to provide freedom of motion in the 360° direction N joints are provided and these are attached with the processor in order to provide movement in particular direction. CORDIC processors processes the task by taking the input in the form of coordinates such that each joint is indicated by the coordinates (x_i, y_i, z_i) where $i \in \{0,1,2, \ldots \ldots, N-1\}$. The rotational and translational motion both are evaluated by summing up of coordinate values.

Various robotic controls that uses CORDIC are suggested in [61], [62] and it serves as the heart of the device. CORDIC has also been used for obstacle avoidance and for determining the location of the dynamic objects as shown in [63], [64], [65]. Dynamic obstacle detection involves the multiple transformations of the coordinates. Thus CORDIC architecture can perform these tasks with only simple circuitry and using only shift and add operations.

CORDIC has also helped in graphical interfaces such as it deals with the coordinate rotation, pixel rotation, contrast and interpolation of geometric components of graphic. 3D graphics that can be implemented using efficient framework has been simply represented using the CORDIC architecture in [66], [67]. For enhancing the great features of shading and contrast in graphics, vector interpolation is implemented using CORDIC [68], [69].

Comparison table of existing approaches using vector rotation for input width=16

Table 1				
	Hardware required	**Full Adder count**	**Latency**	**Latency($1T_{CSA} = 9T_{FA}$)**
Direct implementation[1]	4 multiplier+3adders	1072	$17T_{FA}$+$1T_{CSA}$	$26T_{FA}$
Radix-4 CORDIC[11-13]	51 Adders	816	$24T_{FA}$+$1T_{CSA}$	$33T_{FA}$
Angle recoding[14]	47 Adders	752	$22T_{FA}$+$1T_{CSA}$	$31T_{FA}$
EEAS [15-16]	29 Adders	464	$10T_{FA}$+$1T_{CSA}$	$19T_{FA}$
MSR-CORDIC[69]	21 Adders	336	$7T_{FA}$+$1T_{CSA}$	$16T_{FA}$
TCORDIC[70]	-	12956	-	$16T_{FA}$

The latency (speed) of the existing approaches is compared in Table 1. In this table, FA denotes the Full Adder, T_{FA} denotes the delay introduced by a single full adder, and T_{CSA} denotes the delay of Carry Select Adder (CSA). In comparison table, the Carry-Save Adder (CSA) scheme has been applied in all CORDIC architectures, and vector merges with carry select adder at the last stage are performed. It can be seen from the table that MSR CORDIC and TCORDIC show the least latency and have low hardware complexity in comparison to other algorithms.

CONCLUSION

The CORDIC algorithm has led to the tremendous emphasis in the areas of research, signal processing, graphic and wireless communication. It is specifically known for its supercomputing functionality. CORDIC algorithms were designed and evaluated for the purpose of solving the real time navigation hindrances. It provides unique solution set for evaluating trigonometric computations, waveform generation, used for multiplication, square root, division, analog and digital processing, robotics and graphic vector optimization. The importance of CORDIC architecture has led to the enhancement of its features by using parallel and pipelined implementation and also by reducing the number of iterations. It provides framework that is used for rotation of a vector in circular, hyperbolic or linear coordinate systems. It is also used for generation of sine and cosine waveforms, arithmetic operations such as multiplication and division, and rotation of angle, geometric calculations, logarithms, exponentials and square root. Moreover, CORDIC architecture provides us with an ease to evaluate the variety of parameters in signal processing, communication, robotics and graphics. The efficiency of the algorithm has been increased using the different approaches such as parallel and pipeline. Effect of latency that has been the major drawback is reduced using the higher order radix as well as angle recoding techniques. CORDIC architecture has also shown reduction in the hardware complexity with the implementation of the features of pipelining. CORDIC has shown an unexpected impulse in the field of medical and robotics that helps the people in daily navigation as well as help in keeping the head of the patient at particular position during operation. As CORDIC is implemented on field programmable gate array, it is very useful for high throughput and fast speed real time applications. With the improvement in latency and throughput of CORDIC architecture, this framework can be further enhanced for reducing hardware complexity and can be used in high speed applications which can be explored in near future.

References

[1]. J. E. Volder, "The CORDIC trigonometric computing technique," IRE Transactions on Electronic Computers, vol. EC-8, pp. 330–334, Sept. 1959.
[2]. J. S. Walther, "A unified algorithm for elementary functions," inProc. 38th Spring Joint Computer Conference., Atlantic City, NJ, 1971, pp. 379–385.
[3]. D. S. Cochran, "Algorithms and accuracy in the HP-35," Hewlett-Packard J., pp. 1–11, Jun. 1972.
[4]. D. Timmermann, H. Hahn, and B. J. Hosticka, "Low latency time CORDIC algorithms," IEEE Transactions Computers, vol. 41, no. 8, pp. 1010–1015, Aug. 1992.

[5]. S. Wang, V. Piuri, and J. E. E. Swartzlander, "Hybrid CORDIC algorithms," IEEE Transactions Computers, vol. 46, no. 11, pp. 1202–1207, Nov. 1997.
[6]. S. Wang and E. E. Swartzlander, Jr., "Merged CORDIC algorithm," in IEEE International Symposium on Circuits and Systems (ISCAS'95), 1995, vol. 3, pp. 1988–1991.
[7]. B. Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," Microelectronics Journal, vol. 33, pp. 77–89, 2002.
[8]. S. Suchitra, S. Sukthankar, T. Srikanthan, and C. T. Clarke, "Elimination of sign precomputation in flat CORDIC," in IEEE International Symposium on Circuits and Systems, ISCAS'05, May 2005, vol. 4, pp. 3319–3322.
[9]. E. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC architectures for fast VLSI filtering and array processing," in IEEE International Conference on Acoustic, Speech, Signal Processing, ICASSP'84,Mar. 1984, vol. 9, pp. 250–253.
[10]. H. Kunemund, S. Soldner, S. Wohlleben, and T. Noll, "CORDIC processor with carry save architecture," in Sixteenth European Solid-State Circuits Conference, ESSCIRC 90, Sept. 1990, pp. 193–196.
[11]. E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapatai, "High performance rotation architectures based on the radix-4 CORDIC algorithm," IEEE Transactions Computers, vol. 46, no. 8, pp. 855–870, Aug. 1997.
[12]. P. R. Rao and I. Chakrabarti, "High-performance compensation technique for the radix-4 CORDIC algorithm," IEE Proceedings - Computers and Digital Techniques, vol. 149, no. 5, pp. 219–228, Sep. 2002.
[13]. E. Antelo, T. Lang, and J. D. Bruguera, "Very-high radix circular CORDIC: Vectoring and unified rotation/vectoring," IEEE Transactions Computers, vol. 49, no. 7, pp. 727–739, July 2000.
[14]. Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," IEEE Transactions Computers, vol. 42, no. 1, pp. 99–102, Jan. 1993.
[15]. Y. H. Hu and H. H. M. Chern, "A novel implementation of CORDIC algorithm using backward angle recoding (BAR)," IEEE Transactions Computers, vol. 45, no. 12, pp. 1370–1378, Dec. 1996.
[16]. C.-S.Wu, A.-Y.Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 50, no. 9, pp. 589–601, Sep. 2003.
[17]. M. Kuhlmann and K. K. Parhi, "P-CORDIC: A precomputation based rotation CORDIC algorithm," EURASIP Journal on Applied Signal Processing, vol. 2002, no. 9, pp. 936–943, 2002.
[18]. D. Fu and A. N. Willson, Jr., "A high-speed processor for digital sine/ cosine generation and angle rotation," in Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers, Nov. 1998, vol. 1, pp. 177–181.
[19]. C.-Y. Chen and W.-C. Liu, "Architecture for CORDIC algorithm realization without ROM lookup tables," in Proceedings 2003 International Symposium on Circuits and Systems, ISCAS'03, May 2003, vol. 4, pp. 544–547.
[20]. D. Fu and A. N. Willson, Jr., "A two-stage angle-rotation architecture and its error analysis for efficient digital mixer implementation," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 53, no. 3, pp. 604–614, Mar. 2006.
[21]. S. Ravichandran and V. Asari, "Implementation of unidirectional CORDIC algorithm using precomputed rotation bits," in 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS 2002, Aug. 2002, vol. 3, pp. 453–456.
[22]. C.-Y. Chen and C.-Y. Lin, "High-resolution architecture for CORDIC algorithm realization," in Proceedings International Conference on Communication, Circuits and Systems, ICCCS'06, June 2006, vol. 1, pp. 579–582.
[23]. D. D. Hwang, D. Fu, and A. N. Willson, Jr., "A 400-MHz processor for the conversion of rectangular to polar coordinates in 0.25-μm CMOS," IEEE Journals on Solid-State Circuits, vol. 38, no. 10, pp. 1771–1775, Oct. 2003.
[24]. S.-W. Lee, K.-S. Kwon, and I.-C. Park, "Pipelined cartesian-to-polar coordinate conversion based on SRT division," IEEE Transactions Circuits and Systems II: Express Briefs, vol. 54, no. 8, pp. 680–684, Aug. 2007.
[25]. D. E. Metafas and C. E. Goutis, "A floating point pipeline CORDIC processor with extended operation set," in IEEE International Symposium on Circuits and Systems, ISCAS'91, June 1991, vol. 5, pp. 3066–3069.
[26]. Z. Feng and P. Kornerup, "High speed DCT/IDCT using a pipelined CORDIC algorithm," in 12th Symposium on Computer Arithmetic, July 1995, pp. 180–187.

[27]. M. Jun, K. K. Parhi, G. J. Hekstra, and E. F. Deprettere, "Efficient implementations of pipelined CORDIC based IIR digital filters using fast orthonormal _-rotations," IEEE Transactions on Signal Processing, vol. 48, no. 9, 2000.
[28]. M. Chakraborty, A. S. Dhar, and M. H. Lee, "A trigonometric formulation of the LMS algorithm for realization on pipelined CORDIC," IEEE Transactions on Circuits and Systems II, Express Briefs, vol. 52, no. 9, 2005.
[29]. E. I. Garcia, R. Cumplido, and M. Arias, "Pipelined CORDIC design on FPGA for a digital sine and cosine waves generator," in International Conference on Electrical and Electronics Engineering, ICEEE'06, Sept. 2006, pp. 1–4.
[30]. J.-M. Muller, Elementary Functions: Algorithms and Implementation. Boston, MA: Birkhauser Boston, 2006.
[31]. Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," IEEE Signal Processing Magzine, vol. 9, no. 3, pp. 16–35, July 1992.
[32]. F. Angarita, A. Perez-Pascual, T. Sansaloni, and J. Vails, "Efficient FPGA implementation of cordic algorithm for circular and linear coordinates," in International Conference on Field Programmable Logic and Applications, Aug. 2005, pp. 535–538.
[33]. P. K. Meher, J. K. Satapathy, and G. Panda, "Efficient systolic solution for a new prime factor discrete Hartley transform algorithm," Proceedings IEE Circuits, Devices & Systems, vol. 140, no. 2, pp. 135–139, Apr. 1993.
[34]. L.-W. Chang and S.-W. Lee, "Systolic arrays for the discrete Hartley transform," IEEE Transactions Signal Processing, vol. 39, no. 11, pp. 2411–2418, Nov. 1991.
[35]. P. K. Meher and G. Panda, "Novel recursive algorithm and highly compact semisystolic architecture for high throughput computation of 2-D DHT," Electronics Letters, vol. 29, no. 10, pp. 883–885, May 1993.
[36]. W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," IEEE Transactions on Communications, vol. 25, no. 9, pp. 1004–1009, Sep. 1977.
[37]. B. Das and S. Banerjee, "Unified CORDIC-based chip to realise DFT/ DHT/DCT/DST," Proc. IEE Computers and Digital Techniques, vol. 149, no. 4, pp. 121–127, July 2002.
[38]. J.-H. Hsiao, L.-G. Ghen, T.-D. Chiueh, and C.-T. Chen, "High throughput CORDIC-based systolic array design for the discrete cosine transform," IEEE Transactions on Circuits and Systems Video Technology, vol. 5, no. 3, pp. 218–225, June 1995.
[39]. D. C. Kar and V. V. B. Rao, "A CORDIC-based unified systolic architecture for sliding window applications of discrete transforms," IEEE Transactions on Signal Processing, vol. 44, no. 2, pp. 441–444, Feb. 1996.
[40]. Hong-Thu Nguyen, Xuan-Thuan Nguyen, Cong-Kha Pham, "An efficient fixed-point arithmetic processor using a hybrid CORDIC algorithm", 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 22 February 2018.
[41]. A. M. Despain, "Fourier transform computers using CORDIC iterations," IEEE Transactions on Computers, vol. 23, no. C-10, pp. 993–1001, Oct. 1974.
[42]. K. J. Jones, "High-throughput, reduced hardware systolic solution to prime factor discrete Fourier transform algorithm," Proceedings IEE Computers and Digital Technology, vol. 137, no. 3, pp. 191–196, May 1990.
[43]. K. J. Jones, "2D systolic solution to discrete Fourier transform," Proceedings IEE Computers and Digital Technology, vol. 136, no. 3, pp. 211–216, May 1989.
[44]. T.-Y. Sung, "Memory-efficient and high-speed split-radix FFT/IFFT processor based on pipelined CORDIC rotations," Proceedings IEE Vision, Image Signal Processing, vol. 153, no. 4, pp. 405–410, Aug. 2006.
[45]. B. Das and S. Banerjee, "Unified CORDIC-based chip to realise DFT/ DHT/DCT/DST," Proceedings IEE Computers and Digital Techniques, vol. 149, no. 4, pp. 121–127, July 2002.
[46]. J.-H. Hsiao, L.-G. Ghen, T.-D. Chiueh, and C.-T. Chen, "High throughput CORDIC-based systolic array design for the discrete cosine transform," IEEE Transactions on Circuits and Systems Video Technology, vol. 5, no. 3, pp. 218–225, June 1995.
[47]. D. C. Kar and V. V. B. Rao, "A CORDIC-based unified systolic architecture for sliding window applications of discrete transforms," IEEE Transactions on Signal Processing, vol. 44, no. 2, pp. 441–444, Feb. 1996.
[48]. Jian Wang, Chunlin Xiong, Kangli Zhang, Jibo Wei, "Fixed-Point Analysis and Parameter Optimization of the Radix-**2k** Pipelined FFT Processor", IEEE Transactions on Signal Processing, Volume: 63, Issue: 18 , Sept.15, 2015.

[49]. Y. H. Hu and S. Naganathan, "A novel implementation of chirp Z-transform using a CORDIC processor," IEEE Transactions on Acoustics, Speech, Signal Processing, vol. 38, no. 2, pp. 352–354, Feb. 1990.
[50]. R. C. Gonzalez, Digital Image Processing, 3rd ed. Upper Saddle River, N.J.: Prentice Hall, 2008.
[51]. N. Guil, J. Villalba, and E. L. Zapata, "A fast Hough transform for segment detection," IEEE Transactions on Image Processing, vol. 4, no. 11, pp. 1541–1548, Nov. 1995.
[52]. S. M. Bhandakar and H. Yu, "VLSI implementation of real-time image rotation," in International Conference on Image Processing, Sept. 1996, vol. 2, pp. 1015–1018.
[53]. S. Sathyanarayana, S. R. Kumar, and S. Thambipillai, "Unified CORDIC based processor for image processing," in 15th International Conference on Digital Signal Processing, July 2007, pp. 343–346.
[54]. Yafeng Yao and Zhongxiu Feng, "BBR-Based Iteration-Free CORDIC Algorithm", Journal of Circuits, Systems and Computers, Volume 27, No. 05, September 2018.
[55]. Tushar Supe, David Anderson, "Super-CORDIC: An approximation based parallel and redundant CORDIC algorithm", 2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), October 2016.
[56]. L. Cordesses, "Direct digital synthesis: A tool for periodic wave generation (part 1)," IEEE Signal Processing Magzine, vol. 21, no. 4, 2004.
[57]. Dongyuan Shi, Zishu He, "A new frequency source based on Sigma Delta and CORDIC", 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), 17 May 2012.
[58]. J. Vankka, "Methods of mapping from phase to sine amplitude in direct digital synthesis," in 50th IEEE International Frequency Control Symposium, Jun. 1996, pp. 942–950.
[59]. F. Cardells-Tormo and J. Valls-Coquillat, "Optimisation of direct digital frequency synthesisers based on CORDIC," Electronics Letters, vol. 37, no. 21, 2001.
[60]. G.Vishnu, P.Karthik, Fathima Jabeen, "VLSI Design and Implementation of Efficient Software Defined Radio Using Optimized Quadrature Direct Digital Frequency Synthesizer on FPGA", Procedia Computer Science, Volume 58, 2015, Pages 414-421.
[61]. Y.Wang and S. Butner, "A new architecture for robot control," in Proceedings IEEE International Conferences on Robotics and Automation, 1987, pp. 664–670.
[62]. Y. Wang and S. Butner, "RIPS: A platform for experimental real-time sensory-based robot control," IEEE Transactions on Systems, Man, Cybernetics, vol. 19, pp. 853–860, 1989.
[63]. M. Kameyama, T. Amada, and T. Higuchi, "Highly parallel collision detection processor for intelligent robots," IEEE Journals on Solid-State Circuits, vol. 27, pp. 300–306, 1992.
[64]. Linbin Chen ; Jie Han ; Weiqiang Liu ; Fabrizio Lombardi, "Algorithm and Design of a Fully Parallel Approximate Coordinate Rotation Digital Computer (CORDIC)", IEEE Transactions on Multi-Scale Computing Systems, Volume: 3 , Issue: 3 , July-Sept. 1 2017.
[65]. Pranjal Vyas, Leena Vachhani, K. Sridharan, Vikramkumar Pudi, "CORDIC-Based Azimuth Calculation and Obstacle Tracing via Optimal Sensor Placement on a Mobile Robot", IEEE/ASME Transactions on Mechatronics, Volume: 21 , Issue: 5 , pp-2317 – 2329, Oct. 2016.
[66]. T. Lang and E. Antelo, "High-throughput CORDIC-based geometry operations for 3D computer graphics," IEEE Transactions on Computers, vol. 54, no. 3, pp. 347–361, Mar. 2005.
[67]. Rajkumar Ramadoss, Mehran Mozaffari Kermani, "Reliable Hardware Architectures of the CORDIC Algorithm with a Fixed Angle of Rotations", IEEE Transactions on Circuits and Systems II: Express Briefs, Volume: 64, Issue: 8 , Aug. 2017.
[68]. Suraj N. Shinde, "Twiddle factor generation using CORDIC processor for fingerprint application", International Conference on Computer, Communication and Control (IC4), Sept. 2015.
[69]. Chih-Hsiu Lin ; An-Yeu Wu, "Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for high-performance vector rotational DSP applications", IEEE Transactions on Circuits and Systems I: Regular Papers, pp: 2385 - 2396Volume: 52 , Issue: 11 , Nov. 2005.
[70]. Baozhou Zhu, Yuanwu Lei, Yuanxi Peng, and Tingting He, "Low Latency and Low Error Floating-Point Sine/Cosine Function Based TCORDIC Algorithm", IEEE Transactions on Circuits and Systems I: Regular Papers, Volume: 64, Issue: 4 , April 2017.