CORDIC Algorithm and its Implementation

January 5, 2019

1 Introduction

COordinate Rotation Digital Computer (CORDIC) is a simple and efficient algorithm to compute arithmetic, trigonometric and hyperbolic functions. In many fields such as DSP, image processing, communication or in industrial sectors, researchers are using CORDIC algorithm to optimize design performance. CORDIC was conceived in 1956 by Jack E. Volder thus it sometimes called as Volder's algorithm. Invention of CORDIC paved the way for computing several functions by same hardware in an iterative fashion. Later CORDIC algorithm is polished and optimized by several researchers. In this tutorial we will discuss about basic theory and its hardware implementation.

2 Theoretical Background

In this section the basic theory behind the CORDIC algorithm is discussed. At the end of this discussion we will formulate the basic equations of CORDIC algorithm. The diagram in Figure 1 shows three points (x, y), (x_1, y_1) and (x_2, y_2) on a circular path in the x-y co-ordinate system. As the points are on the circular path, distance of all the points from the origin is the same which is r here. Let us define our objective,which is to rotate the point (x, y) anticlockwise towards the point (x_2, y_2).

For the point (x, y) following equations can be written

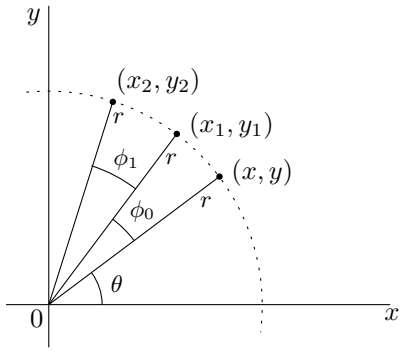

Figure 1: Rotation on Circular path

$$x = rcos\theta \text{ and } y = rsin\theta \tag{1}$$

and for the point (x_1, y_1) following equations can be written

$$x_1 = rcos(\theta + \phi_0) \text{ and } y_1 = rsin(\theta + \phi_0) \tag{2}$$

To rotate the point (x, y) to the point (x_2, y_2) total angle to be rotated anticlockwise is $(\phi_0+\phi_1)$. At the first step, point (x, y) will be rotated by angle ϕ_0 in anticlockwise direction to reach the point (x_1, y_1). The expression of x_1 and y_1 in terms of points x and y can be expressed as follows.

$$\begin{aligned} x_1 &= rcos(\theta + \phi_0) && (3)\\ &= rcos\theta cos\phi_0 - rsin\theta sin\phi_0 && (4)\\ &= xcos\phi_0 - ysin\phi_0 && (5)\\ &= cos\phi_0(x - ytan\phi_0) && (6) \end{aligned}$$

Similar equations can be written for y also

$$\begin{aligned} y_1 &= rsin(\theta + \phi_0) && (7)\\ &= rsin\theta cos\phi_0 + rcos\theta sin\phi_0 && (8)\\ &= ycos\phi_0 + xsin\phi_0 && (9)\\ &= cos\phi_0(y + xtan\phi_0) && (10) \end{aligned}$$

Initial angle to be rotated in anticlockwise direction was $(\phi_0 + \phi_1)$ which is equal to z_0. The next angle to be rotated can be expressed as

$$z_1 = z_0 - \phi_0 = \phi_1 \quad (11)$$

In the next, we will rotate the point (x_1, y_1) to the point (x_2, y_2). The expressions for x_2 and y_2 in terms of x and y is expressed as follows

$$\begin{aligned} x_2 &= rcos(\theta + \phi_0 + \phi_1) && (12)\\ &= rcos(\theta + \phi_0)cos\phi_1 - rsin(\theta + \phi_0)sin\phi_1 && (13)\\ &= x_1 cos\phi_1 - y_1 sin\phi_1 && (14)\\ &= cos\phi_1(x_1 - y_1 tan\phi_1) && (15) \end{aligned}$$

Similar equations can be written for y also

$$\begin{aligned} y_2 &= rsin(\theta + \phi_0 + \phi_1) && (16)\\ &= rsin(\theta + \phi_0)cos\phi_1 + rcos(\theta + \phi_0)sin\phi_1 && (17)\\ &= y_1 cos\phi_1 + x_1 sin\phi_1 && (18)\\ &= cos\phi_1(y_1 + x_1 tan\phi_1) && (19) \end{aligned}$$

and the equation for the remaining angle is

$$z_2 = z_1 - \phi_1 = 0 \quad (20)$$

Now we are able to formulate the general expression of x_i and y_i after rotation by certain angle in anticlockwise direction. The general expressions are

$$\begin{aligned} x_i &= cos\phi_{i-1}(x_{i-1} - y_{i-1}tan\phi_{i-1}) && (21)\\ y_i &= cos\phi_{i-1}(y_{i-1} + x_{i-1}tan\phi_{i-1}) && (22)\\ z_i &= z_{i-1} - \phi_{i-1} && (23) \end{aligned}$$

If the above expressions of x_i and y_i is written in terms of initial point x and y then the expressions will be

$$x_i = \prod_{j=0}^{i-1} cos\phi_j(.....) \quad (24)$$

$$y_i = \prod_{j=0}^{i-1} cos\phi_j(.....) \quad (25)$$

In every iteration i, a constant term is associated with the equation of x_i and y_i. For the shake of easy implementation, the computation of x_i and y_i is done without the constant term. The final values at the output stage x_f and y_f are divided by a constant term k_n to obtain the actual results. The expression of k_n is derived below.

The term $\prod_{j=0}^{i-1} cos\phi_j$ is a constant and need not be evaluated at each iteration. The value of this constant can be evaluated as follows

$$\prod_{j=0}^{i-1} cos\phi_j = \prod_{j=0}^{i-1} \frac{cos\phi_j}{\sqrt{cos^2\phi_j + sin^2\phi_j}} \tag{26}$$

$$= \frac{1}{\prod_{j=0}^{i-1} \sqrt{1 + tan^2\phi_j}} \tag{27}$$

$$= \frac{1}{\prod_{j=0}^{i-1} \sqrt{1 + 2^{-2j}}} = \frac{1}{k_n} \tag{28}$$

Angle representation is very important in this case as every iteration corresponds to an incremental rotation. The most used angle rotation for data width of m is

$$-\pi \ \frac{\pi}{2^0} \ \frac{\pi}{2^1} \ \frac{\pi}{2^2} \ \frac{\pi}{2^3} \ \frac{\pi}{2^4} \ldots\ldots \frac{\pi}{2^m} \tag{29}$$

This format is suitable to represent any angle. The two MSB bits represents the location of the co-ordinate in any quadrant. For example, angle of 45 degree can be represented in 16 bit data format as 16'b0010000000000000, which is in the first quadrant.

So the angles are represented in terms of power of 2. Let's put $tan\phi_{i-1} = 2^{i-1}$ and the above equations become

$$x_i = x_{i-1} - y_{i-1}2^{i-1} \tag{30}$$

$$z_i = y_{i-1} + x_{i-1}2^{i-1} \tag{31}$$

$$z_i = z_{i-1} - tan^{-1}2^{i-1} \tag{32}$$

The computation of the above equations become easy now as it involves division by power of 2 and addition or subtraction. Division by power of 2 is performed by wired shift method which is hardware less. Only hardware is required is an adder or a subtractor.

The above example is based on the fact that rotation is done only in anticlockwise direction. Actually the target angle rotation is achieved by successive incremental rotations until the difference between the target angle and achieved angle becomes zero. They are called micro rotations and can be anticlockwise or clockwise. These situation is described Figure 2.

The sign of angle difference (z_i), decides the direction of the next micro rotation and a new parameter σ is introduced.

$$\sigma_i = \begin{cases} 1, & \text{if } z_i \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{33}$$

Another point is that these micro rotations do not exactly follow the circular path as the constant term $cos\phi_{i-1}$ is associated with each micro rotation. For the simplicity of calculation, the constant term is not calculated in each iteration. Computation is done without the constant term and after the final iteration, computed results are divided by the constant term. Thus the modified equations are.

$$x_i = x_{i-1} - \sigma_i y_{i-1}2^{i-1} \tag{34}$$

$$z_i = y_{i-1} + \sigma_i x_{i-1}2^{i-1} \tag{35}$$

$$z_i = z_{i-1} - \sigma_i tan^{-1}2^{i-1} \tag{36}$$

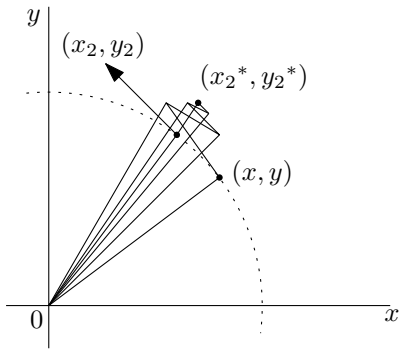

Figure 2: CORDIC Micro-rotations

After the above discussion we have formulated the equations for rotating a point (x, y) by any angle on circular path. Thus CORDIC can be used to rotate a point (x, y) in any direction.

We have discussed rotation by maximum 90 degree in any direction. But if the rotation by more than 90 degree is to be achieved then what will be the technique. Actually by trigonometric rules, rotation by other angles can be realized in terms of rotation by 90 degree. This is explained below in Table 1. If a point lies in 2nd or 3rd co-ordinate, rotation is achieved by assuming that it lies in the 1st or 4th co-ordinate. Actual result is obtained by conditional sign change at the output. This is explained with Figure 3. To rotate a point lies in the 2nd quadrant, (initial angle θ lies between $\pi/2 \leq \alpha < \pi$) the point is assumed to be in 4th quadrant and rotation is done as if the point lies in 4th quadrant. At the output stage, output is inverted to obtain actual result.

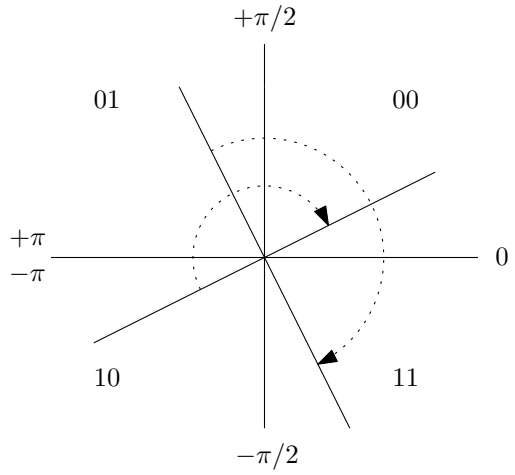

Figure 3: Quadrant transformation for rotation on circular path

Table 1: Quadrant transformation and sign change for CORDIC

Before Transformation			After Transformation		
$\theta[15]\theta[14]$	Range	Quadrant	$\theta[15]\theta[14]$	Quadrant	Sign Change at O/P
00	$0 \leq \theta < \pi/2$	1st	00	1st	no
01	$\pi/2 \leq \theta < \pi$	2nd	11	4th	yes
10	$-\pi \leq \theta < -\pi/2$	3rd	00	1st	yes
11	$-\pi/2 \leq \theta < 0$	4th	11	4th	no

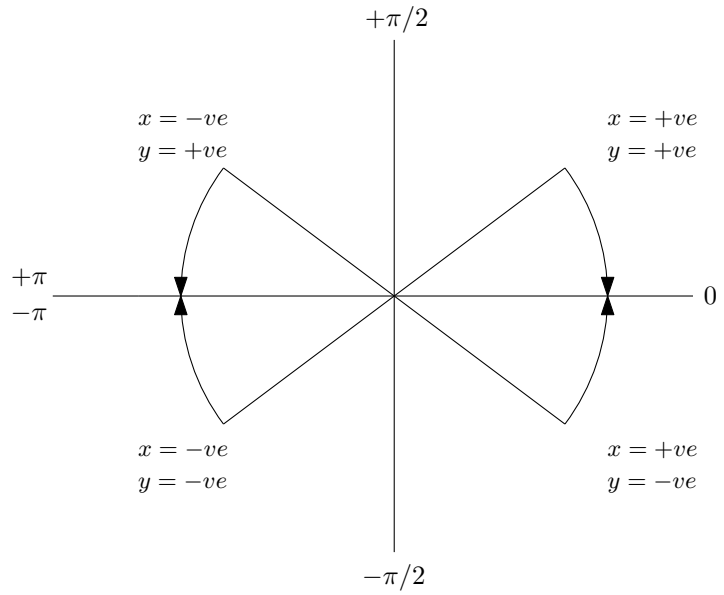

Figure 4: Vectoring Operation

3 Vectoring Mode

Another situation may arrive where a co-ordinate is given and we have to find the angle. This situation is simply the opposite of the above mentioned Rotation mode. This is called the Vectoring mode, In this mode, one of the co-ordinate is nullified and we get the angle between them at the output. The general equations for this mode is described below.

$$x_i = x_{i-1} + \sigma_i y_{i-1} 2^{i-1} \tag{37}$$
$$z_i = y_{i-1} - \sigma_i x_{i-1} 2^{i-1} \tag{38}$$
$$z_i = z_{i-1} + \sigma_i tan^{-1} 2^{i-1} \tag{39}$$

Generally the co-ordinate y is nullified the final expressions are $x_f = K_f\sqrt{x^2 + y^2}$ and $z_f = tan^{-1}(y/x)$. So at the output we get the magnitude and phase. This mode is very useful to compute the absolute of (x,y) or to nullify one of the co-ordinate. The value of the constant term is same as mentioned above. The equation of σ is different here and it is mentioned below.

$$\sigma_i = \begin{cases} 1, & \text{if } y_i \leq 0 \\ -1, & \text{otherwise} \end{cases} \tag{40}$$

The co-ordinates x and y, can be +ve or -ve and depending on their sign, the sign of the final output and computed angle is needed to be modified accordingly at the output. This situation is explained below with Figure 4 and Table 3.

3.1 Computation of Sine and Cosine

The general expression of the CORDIC algorithm can be expressed as

$$x_i = k_n(x_{i-1} cos\phi_{i-1} - y_{i-1} sin\phi_{i-1}) \tag{41}$$
$$y_i = k_n(y_{i-1} cos\phi_{i-1} + x_{i-1} sin\phi_{i-1}) \tag{42}$$
$$z_i = z_{i-1} + \sigma_i tan^{-1} 2^{i-1} \tag{43}$$

Table 2: Sign change for vectoring operation

x[15]y[15]	Rotation	Sign Change (O/P)
00	clockwise	no
01	anticlockwise	no
10	anticlockwise	yes
11	clockwise	yes

In rotation mode, if the initial value of input y is set to zero and the initial value of input x is set to $\frac{1}{k_n}$ then

$$x_1 = cos\phi_0 \tag{44}$$
$$y_1 = sin\phi_0 \tag{45}$$
$$z_1 = z_{in} + \sigma_i tan^{-1}2^0 \tag{46}$$

In the second iteration

$$x_2 = cos\phi_0 cos\phi_1 - sin\phi_0 sin\phi_1 = cos(\phi_0 + \phi_1) \tag{47}$$
$$y_2 = sin\phi_0 sin\phi_1 + cos\phi_0 sin\phi_1 = sin(\phi_0 + \phi_1) \tag{48}$$
$$z_2 = z_1 + \sigma_i tan^{-1}2^1 \tag{49}$$

This way The final outputs of CORDIC x_f and y_f holds the value $cosz_{in}$ and $sinz_{in}$ respectively.The initial angle is z_{in}. The value σ is computed as per value of z.

4 Linear Mode

Rotation of the co-ordinates can also be done in linear path. Rotation on linear path is described in Figure 5. Final equation of the co-ordinates can be obtained by the same way as it is done

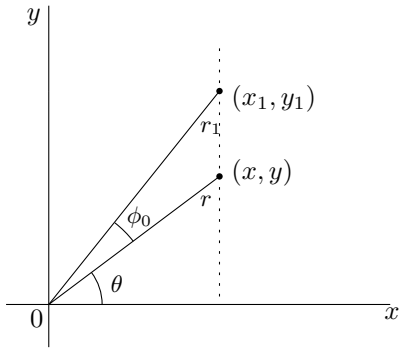

Figure 5: Rotation on linear path

above. The final equations for the linear mode are

$$x_i = x_{in} \tag{50}$$
$$y_i = y_{i-1} + \sigma_i x_{i-1} 2^{i-1} \tag{51}$$
$$z_i = z_{i-1} - \sigma_i 2^{i-1} \tag{52}$$

One of the benefit of the linear mode is that the scaling factor $k_n = 1$. Thus error due to scaling factor is eliminated. Two important operations can be performed in this mode which are multiplication and division.

4.1 Multiplication

Multiplication is performed in rotation mode. The value of σ is determined by the same equation as mentioned above. The CORDIC evaluates produces final result as

$$y_f = y_{in} + z_{in} * x_{in} \tag{53}$$

Initially y_{in} is set to zero and y_f holds the final result. Multiplication result is bound by the data width of x,y and z. This is the disadvantage of this method. For 18 bit data-width,multiplication result will also be limited by 18-bit.

4.2 Division

Division is performed in vector mode. The value of σ is determined by the same equation as mentioned above for vectoring mode. The CORDIC evaluates produces final result as

$$z_f = z_{in} + y_{in}/x_{in} \tag{54}$$

Initially Z_{in} is set to zero and z_f holds the division output. The major limitation of the division operation is that the final result is bound to be fit in the data width of input operands. But on the advantage side, CORDIC divider has low latency and consumes less hardware compared to the other fast dividers. Accuracy is also is a concern where high accuracy is needed.

5 Hyperbolic Mode

Similar to circular and linear, rotation of co-ordinates can also be done along a hyperbola. This is an extension of the CORDIC algorithm. Several more functions can be computed using this type of rotation mechanism. The equations can be derived same way as in rotation mode. Rotation on hyperbolic path is explained in the Figure 6. The general expressions for mode are

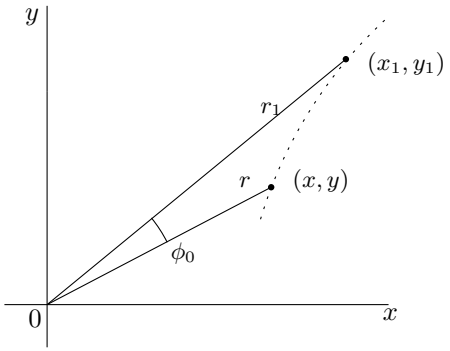

Figure 6: Rotation on hyperbolic path

$$x_i = x_{i-1} + \sigma_i y_{i-1} 2^{i-1} \tag{55}$$

$$z_i = y_{i-1} + \sigma_i x_{i-1} 2^{i-1} \tag{56}$$

$$z_i = z_{i-1} - \sigma_i tanh^{-1} 2^{i-1} \tag{57}$$

The operation in hyperbolic mode is slightly different from the other two modes. In hyperbolic mode, computation starts from iteration 1 as the value $tanh^{-1}2^0 = \infty$. In hyperbolic mode convergence is an issue. To make sure that output will converge, some iterations are repeated. The repetition of the iterations are done by repeating iterations $4, 13, 40, ..., i, 3i + 1, ...$ The scale factor in an iteration i is

$$k_h(i) = (1 - 2^{-2i})^{1/2} \tag{58}$$

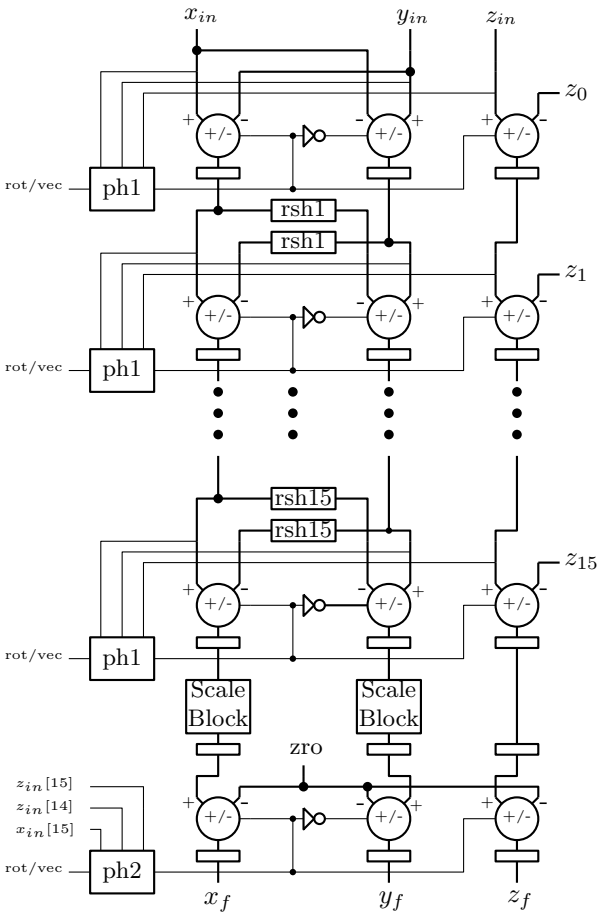

Figure 7: Parallel architecture of CORDIC

5.1 Square Root Computation

To compute square root hyperbolic mode of CORDIC is used. In hyperbolic co-ordinates, the final equation of x and y in vectoring mode are

$$x_f = k_h\sqrt{(x^2 - y^2)} \tag{59}$$

In the above equation,if initial value of $x = a + 1/4$ and $y = a - 1/4$ then it converges to

$$x_f = k_h\sqrt{(4.a.1/4)} = k_h.\sqrt{a} \tag{60}$$

6 Implementation of CORDIC

Invention of CORDIC algorithm opens many scopes of optimization in implementation of digital systems. CORDIC algorithm can compute several functions without any change in hardware. Most direct application of CORDIC is CORDIC based calculator. Nowadays CORDIC is used in many implementations to compute several functions. There are two types of architectures of CORDIC, which are serial and parallel. Implementation of both the architectures are described below.

6.1 Parallel Architecture

The parallel architecture is shown in Figure 7. The architecture is designed by considering 16-bit fixed point data width. The architecture supports both rotation and vector modes. In

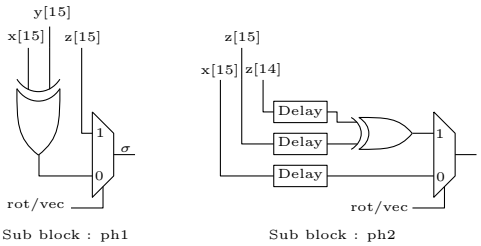

Figure 8: Architecture of phase blocks

Table 3: Iteration wise CORDIC for computation of absolute of x and y

x_i	y_i	z_i	i	σ
8	-2	45	0	1
9	2	18.4349	1	-1
9.5	-0.25	32.4712	2	1
9.5313	0.9375	25.3462	3	-1
9.5898	0.3418	28.9225	4	-1
9.6005	0.0421	30.7124	5	-1
9.6012	-0.1079	31.6076	6	1
9.6020	-0.0329	31.1600	7	1
9.6022	0.0046	30.9362	8	-1
9.6022	-0.0141	31.0481	9	1
9.6022	-0.0048	30.9921	10	1
9.6022	0	30.9641	11	1

the parallel architecture, each iteration corresponds to a stage. Each stage has 3 add/sub units which does addition or subtraction depending on value of signal σ. The value of σ is computed by the $ph1$ block which is shown Figure 8. When σ is 1, add/sub blocks performs subtraction. Different angles for micro rotation are fed to the each stage. The rsh blocks are there for shifting data to the right. These blocks performs wired shifting which is described in detail in *Combinational Curcuits* post. For examples, rsh1 block shifts the data to the right by 1 bit. The scale factor compensation is achieved by using a scale factor. The scale block divides the output of the last stage by k_n by constant multiplication technique. This block is also described in *Combinational Curcuits* post. At the last, a inversion stage is added to invert the outputs of the scale block conditionally. The control signal for the add/sub units at the inversion stage is generated by the $ph2$ block which is shown in Figure 8.

6.2 Serial Architecture

The serial architecture is described in Figure 9. The serial architecture is similar to one stage of parallel CORDIC architecture. The scaling and inversion stage is similar to the parallel CORDIC. The architecture is of 16-bit. The different rotational angles are pre-stored in an LUT. Total 16 angles are stored in that LUT and a 4-bit counter is used to fetch those angles. The VRSH block is variable right shift block to shift the input operands. This block is described in detail in the *Combinational Curcuit* post. The fdc block is a controlled register block which stores a data when enable signal is asserted. The control signal add/sub is similar to the parallel CORDIC. Total number of iterations required to evaluate a function depends on the data-width and data precision.

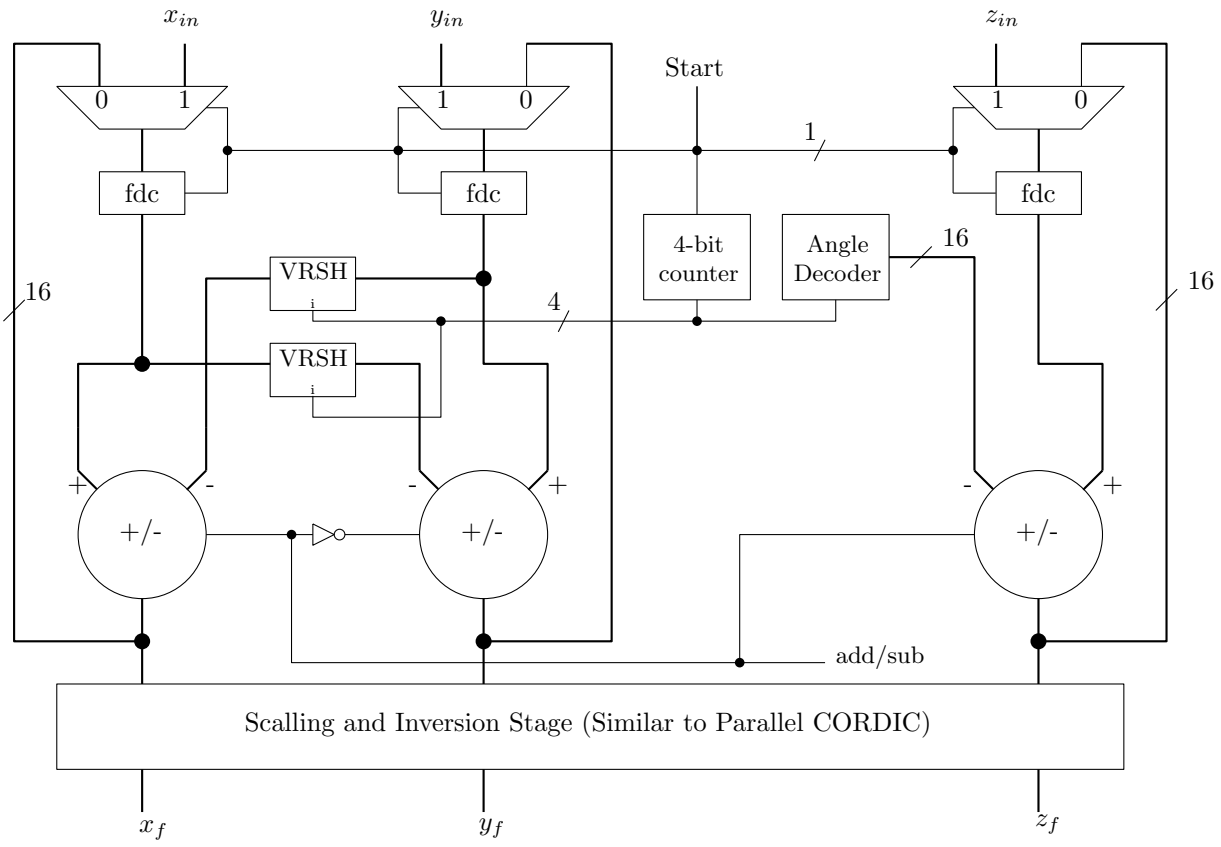

Figure 9: Serial architecture of CORDIC

6.3 Example of CORDIC iteration

An example is given below to understand the evaluation of the final result through iterations. Let's take an example of computing absolute value of x and y. This operation can be done in vectoring mode of circular co-ordinates. The iteration wise evaluation is given below in Table 3. Initially $x_{in} = 5$, $y_{in} = 3$, $z_{in} = 0$ and thus $\sigma = -1$. The value of the scale factor (k_n) is 1.64676. The final output (x_f) is obtained by dividing 9.6022 by k_n. So the absolute of x and y is 5.8310 and angle between them is 30.9641.

7 Application

There exists numerous research works where CORDIC is used. Some of the major application areas are mentioned below

- Most direct application of CORDIC is in making calculator where same hardware can be used to calculate several functions.
- In implementations of many complex systems or algorithms CORDIC may be used to evaluate arithmetic functions. Popular use of CORDIC is to find reciprocal of a number or to evaluate division.
- The property,rotation of co-ordinates is used to generate transform domains such as FFT, Wavelet, Curvelet, DCT etc.
- The CORDIC can find magnitude or absolute of two numbers easily in vectoring mode in circular co-ordinates. This property is used in many applications like to matrix using factorization QR decomposition.

- CORDIC can find sine or cosine of an angle by same hardware. This property is used in many DSP applications.