Computing the Satellite's Coordinates using CORDIC Algorithm

Ms. Ankita Sharma * Mrs. Neha Sharma** Mrs. Sukomal
**M. Tech. Scholar, S.G.T. Institute of Engineering & technology, Gurgaon
** Asst. Professor, SGTIET, Gurgaon
*** Asst. Professor, SGTIET, Gurgaon

Abstract
This paper describes the application of the CORDIC Algorithm's to find the coordinates of the satellite in the x-y plane. The mathematical and analytical approach for the CORDIC algorithm implementation is presented here which shows that CORDIC algorithm revolves around the idea of "rotating" the phase of a complex number, by multiplying it by a succession of constant values.

Keywords: *algorithm, azimuth, coordinates, CORDIC, Elevation, Look angles.*

1. INTRODUCTION
The coordinates to which an earth station antenna must be pointed to communicate with the satellite are called the Look Angles. These are most commonly expressed as azimuth (Az) and elevation (El). Azimuth is measured eastwards from the geographic north to the projection of the satellite path on a locally horizontal plane at the earth station. Elevation angle is measured upward from the local horizontal plane at the earth station to the satellite path.

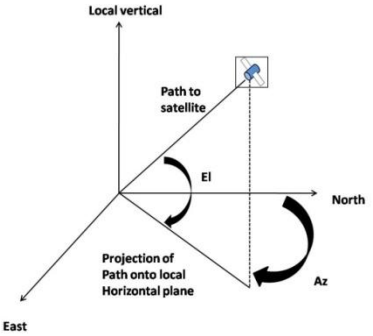

Fig. 1: Look Angles for a Satellite

The name CORDIC is an acronym for Coordinate Rotation Digital Computer. In 1959 Jack E. Volder described the Coordinate Rotation Digital Computer or CORDIC for the calculation of trigonometric functions, multiplication, division and conversion between binary and mixed radix number systems. The CORDIC-algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shift and add.

2. MATHEMATICAL ANALYSIS
General equations for a vector rotation lays the foundation for derivation of the Volder's CORDIC algorithm. If a vector V with coordinates (x, y) is rotated through an angle ϕ then a new vector V' can be obtained with coordinates (x', y') where x' and y' can be obtained using x, y and ϕ by the following method.

$X = r\cos\theta$, $Y = r\sin\theta$ (1)

$(x') = (x.\cos(\phi) - y.\sin(\phi))$ (2)

$(y') = (y.\cos(\phi) + x.\sin(\phi))$ (3)

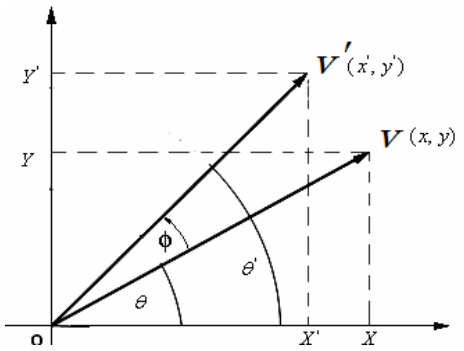

Fig. 2: Rotation of a vector V by angle ϕ

$x' = \cos(\phi)[x - y.\tan(\phi)]$ (4)

$y' = \cos(\phi)[y + x.\tan(\phi)]$ (5)

The multiplication by the tangent term can be avoided if the rotation angles and therefore $\tan(\phi)$ are restricted so that $\tan(\phi) = 2^{-i}$ .In digital hardware this denotes a simple shift operation. Furthermore, if those rotations are performed iteratively and in both directions every value of $\tan(\phi)$ is presentable with $\phi = \arctan(2^{-i})$ the cosine term could also be simplified and since $\cos(\phi) = \cos(-\phi)$ it is a constant for a fixed number of iterations. This iterative rotation can now be expressed as:

$x_{i+1} = k_i [x_i - y_i.d_i.2^{-i}]$ (6)

$y_{i+1} = k_i [y_i - x_i.d_i.2^{-i}]$ (7)

where, i denotes the number of rotation required to reach the required angle of the required vector, $\cos(\arctan(2^{-i}))$ and $d_i = \pm 1$. The product of the K_i's represents the so-called K factor :

$k = \prod_{i=0}^{n-1} k_i$ (8)

Where $\prod_{i=0}^{n-1} k_i = \cos\phi_0 \cos\phi_1 \cos\phi_2 \cos\phi_3 \cos\phi_4$ .....$\cos\phi_{n-1}$(ϕ is the angle of rotation here for n times rotation).

Table 1: Values of Angles for 8-bit CORDIC H/W

i	$d^{-i} = 2^{-i} = tan\phi_i$	ϕ_i= arctan(2^{-i})	ϕ in radian
0	1	45^o	0.7854
1	0.5	26.565^o	0.4636
2	0.25	14.036^o	0.2450
3	0.125	7.125^o	0.1244
4	0.0625	3.576^o	0.0624
5	0.03125	1.7876^o	0.0312
6	0.015625	0.8938^o	0.0156
7	0.0078125	0.4469^o	0.0078

K_i is the gain and its value changes as the number of iteration increases. For 8-bit hardware CORDIC approximation method the value of k_i as

$k_i = \prod_{i=0}^{7} \cos\phi_i = \cos\phi_0 . \cos\phi_1 . \cos\phi_2 . \cos\phi_3 . \cos\phi_4 . \cos\phi_5 . \cos\phi_6 . \cos\phi_7$

=cos45°.cos26.565°.. cos0.04469° = 0.6073 (9)

From the above table it can be seen that precision up to 0.4469° is possible for 8-bit CORDIC hardware.

To simplify each rotation, picking α_i (angle of rotation in *i*th iteration) such that $\alpha_i = d_i . 2\text{-}i$. d_i is such that it has value +1 or -1 depending upon the rotation i. e. $d_i \in \{+1,-1\}$. Then

$x_{i+1} = x_i - d_i y_i 2^i$ (9)

$y_{i+1} = y_i - d_i x_i 2^{-I}$ (10)

$z_{i+1} = z_i - d_i tan^{-1}2^{-I}$ (11)

The computation of x_{i+1} or y_{i+1} requires an i-bit right shift and an add/subtract. If the function $\tan^{-1}2^{-i}$ is pre computed and stored in table (Table 3.1) for different values of i, a single add/subtract suffices to compute z_{i+1}. Each CORDIC iteration thus involves two shifts, a table lookup and three additions. If the rotation is done by the same set of angles (with + or - signs), then the expansion factor K, is a constant, and can be pre computed. For example to rotate by 30 degrees, the following sequence of angles be followed that add up to ≈ 30 degree.

30.0 ≈ □ 45.0- 26.6 + 14.0 - 7.1 + 3.6 + 1.8 - 0.9 + 0.4 - 0.2 + 0.1

= 30.1

In effect, what actually happens in CORDIC is that z is initialized to 30 degree and then, in each step, the sign of the next rotation angle is selected to try to change the sign of z; that is, d_i = sign (z_i) is chosen, where the sign function is defined to be -1 or +1 depending on whether the argument is negative or non-negative. This is reminiscent of no restoring division. Table 2 shows the process of selecting the signs of the rotation angles for a desired rotation of +30 degree.

Table 2: Choosing the signs of the rotation angles to force z to zero

I	z_i -α_i	z_{i+1}
0	+30.0 - 45.0	-15
1	- 15.0 + 26.6	11.6
2	+ 11.6 - 14.0	-2.4
3	- 2.4 + 7.1	4.7
4	+ 4.7 - 3.6	1.1
5	+ 1.1 - 1.8	-0.7
6	- 0.7 + 0.9	0.2
7	+ 0.2 - 0.4	-0.2
8	- 0.2 + 0.2	0
9	+ 0.0 - 0.1	-0.1

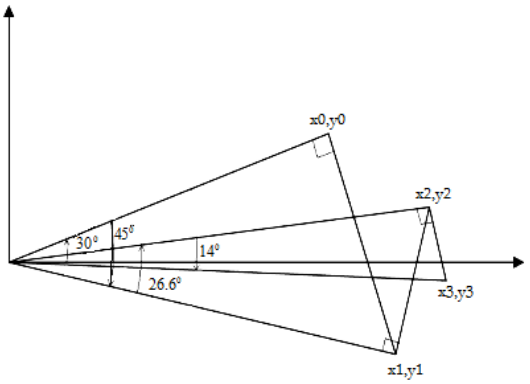

Figure 3: First three of 10 iteration leading from (x_0, y_0) to (x_1, y_1) in rotating by +30°, rotation mode.

3. ALGORITHMIC APPROACH

CORDIC can be used to calculate a number of different functions. This explanation shows how to use CORDIC in rotation mode to calculate sine and cosine of an angle, and assumes the desired angle is given in radians and represented in a fixed point format. To determine the sine or cosine for an angle β, the y or x coordinate of a point on the unit circle corresponding to the desired angle must be found. Using CORDIC, we would start with the vector v_o:

v_o= [1,0] (12)

In the first iteration, this vector would be rotated 45° counterclockwise to get the vector v_1. Successive iterations will rotate the vector in one or the other direction by size decreasing steps, until the desired angle has been achieved. Step i size is arctan $(1/(2^{i-1}))$ for $i = 1, 2, 3, \ldots$.

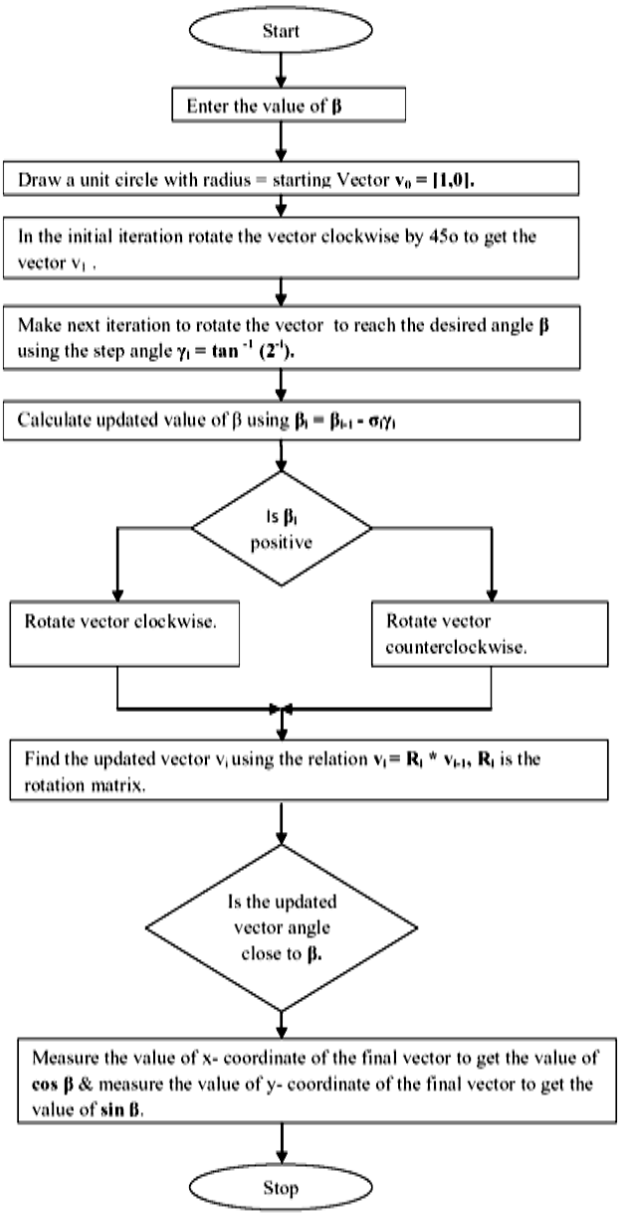

Fig 4: Flowchart for CORDIC Algorithm

4. CONCLUSION

This paper presents mathematical and analytical aspects of implementing the CORDIC algorithm for satellite's communication. For sinusoidal angle calculations in look angle computations this method can provide n-bit accuracy for n iterations. This method is much simpler than the conventional methods as the computations are reduced to only shift and add operations instead of the complex multiplication operations. On the other hand, when a hardware multiplier is available (e.g. in a DSP microprocessor), table-lookup methods and power series are generally faster than CORDIC.

REFERENCES

[1] Ayansola, O.D, Mathematical modelling of antenna look angles of geostationary communications satellite using two models of control stations IEEE Conference Publications- 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), 2010

[2] Ogundele, D.A. Model validation and analysis of antenna look angles of geostationary satellite. IEEE Conference Publications . International Conference on Computer Science and Automation Engineering (CSAE), 2012

[3] R.Andraka. A Survey of CORDIC Algorithms for FPGA Based Computersn – Proc. Of the 1998 CM/SIGDA Sixth International Symposium on FPGAs, February1998, Monterey, CA, pp.191-200.

[4] P.W.Baker. Suggestion for a Binary Cosine Generator, IEEE Transactions on Computers, February, 1975, pp. 1134-1136.

[5] M.Marx. FPGA Implementation of sin(x) and cos(x) Generators Using the CORDIC Algorithm, Final Year Project Report, School of Electronic Engineering, University of Surrey, Guidford, UK, 1999. s

[6] P.Pirsch. Architectures for Digital Signal Processing, John Wiley & Sons,1998.

[7] N.Takagi. Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation, IEEE Trans. On Comput., vol. 40, n 9, 1991, pp. 989-994.

[8] D.Timmerman, H.Hahn, B.J.Hosticka, B.Rix. A New Addition Scheme and Fast Scaling Factor Compensation Methods for CORDIC algorithms, Integration – the VLSI Journal, vol. 11, n 1, 1991, pp. 85-100.

[9] A.Vlachos. Design and Implementation of CORDIC Modules for ADCS, MSc Project Report, School of Electronic Engineering, University of Surrey, Guidford, UK, 1999.

[10] J.Volder. The CORDIC Computing Technique, IRE Trans. Comput., Sept.1959, pp.330-334.

[11] J.S. Walther. A Unified Algorithm for Elementary Functions, Proc. AFIPS Spring Joint Computer Conference, pp.379-385, 1971.