Int. J. Adv. Res. Sci. Technol. Volume 4, Issue 2, 2015, pp.289-293

IJARST

International Journal of Advanced Research in Science and Technology

journal homepage: www.ijarst.com

ISSN 2319 – 1783 (Print)

ISSN 2320 – 1126 (Online)

Implementation of bit serial CORDIC for Robotic Applications.

T. Chandrika and D.P.Raju*

Department of ECE, Koushik college of Engg.,Visakapatnam, Andhra Pradesh, India.

****Corresponding Author's Email: chandrika.tota@gmail.com, dpraju009@gmail.com***

A R T I C L E I N F O

Article history:

Received 20 Nov. 2014

Accepted 11 Dec. 2014

Available online 13 Feb. 2015

Keywords:

CORDIC (coordinate rotation digital computer)

BSI(Bit Serial Iterative)

A B S T R A C T

CORDIC stands for coordinate rotation digital computer. The key concept of CORDIC arithmetic is based on the simple and ancient principles of 2-D geometry. This iterative formulation of a computational algorithm for its implementation is developed for the computation of trigonometric functions and multiplication. Not only a wide variety of applications of CORDIC have been suggested over the time, but also a lot of progress has taken place in the area of algorithm design and development of architectures for high-performance and low-cost solutions .

Rotation of vectors through fixed and known angles has wide applications in robotics, digital signal processing, graphics, games, and animation. Therefore, in this paper, we be present Bit Serial Iterative (BSI) CORDIC Implementation for The Calculation of Trigonometric Functions. Synthesis of the proposed CORDIC Circuit was implemented in xilinx and shown that the proposed design offer higher throughput, less latency for fixed angles of rotation.

Introduction:

CORDIC stands for Coordinate Rotation Digital Computer. It calculates the value of trigonometric functions like sine, cosine, magnitude and phase (arctangent) to any desired precision. It can also calculate hyperbolic functions (such as sinh, cosh, and tanh). The CORDIC works in two different modes, they are Rotation Mode and Vector Mode. In Rotation Mode the algorithm rotates a vector with a given angle, and in Vector Mode the angle between a given vector and the x-axis is calculated. In this project to find trigonometric functions CORDIC works in rotation mode only. The algorithm is multiplication-free and thus well suited for hardware implementation.The CORDIC algorithm does not use Calculus based methods such as polynomial or rational function approximation.

Today cordic algorithm is used in Neural Network VLSI design, high performance vector rotation DSP applications, advanced circuit design, optimized low power design. CORDIC algorithm revolves around the idea of "rotating" the phase of a complex number, by multiplying it by a succession of constant values.

However, the "multiplies" can all be powers of 2, so in binary arithmetic they can be done using just shifts and adds; no actual "multiplier" is needed thus it simpler and do not require complex hardware structure as in the case of multiplier.

Earlier methods used are Table look up method, Polynomial approximation method etc. for evaluation of trigonometric functions. It is hardware efficient algorithm. No multiplier requirement as in the case of microcontroller.

The drawback in CORDIC is that after completion of each iteration, there is a gain which is added to the magnitude of resulting vector which can be removed by multiplying the resulting magnitude with the inverse of the gain. Both rotation mode and vectoring mode methods initialize the angle accumulator with the desired angle value.

CORDIC is generally faster than other approaches when a hardware multiplier is unavailable (e.g. in a microcontroller), or when the number of gates required to implement is to be minimized (e.g. in an FPGA). 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. Various CORDIC architectures like bit parallel iterative CORDIC, a bit parallel unrolled CORDIC, a bit-serial iterative CORDIC and the comparison of various CORDIC architecture has been discussed.

It can be seen that CORDIC is a feasible way to approximate cosine and sine. CORDIC is useful in designing computing devices. As it was originally designed for hardware applications, there are features that make CORDIC an excellent choice for small computing devices. Since it is an iterative method it has the advantage over the other methods of being able to get better accuracy by doing more iteration, whereas the Taylor approximation and the Polynomial interpolation methods need to be rederived to get better results.

These properties, in addition to getting a very accurate approximation is perhaps the reason why CORDIC is used in many scientific calculators today. Due to the simplicity of the involved operations the CORDIC algorithm is very well suited for VLSI implementation. However, the CORDIC iteration is not a perfect rotation which would involve multiplications with sine and cosine.

Basic Equation of Cordic Algorithm:

Volder's algorithm is derived from the general equations for a vector rotation. 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.

$$X = r\cos\theta,\ Y = r\sin\theta$$

$$V' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x.\cos(\phi) - y.\sin(\phi) \\ y.\cos(\phi) + x.\sin(\phi) \end{pmatrix}$$

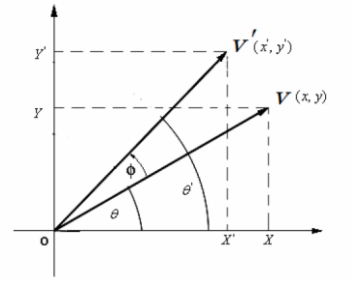

Fig. 1: Rotation of a vector V by the angle Φ

The individual equations for x' and y' can be rewritten as:

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

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

And rearranged so that:

$X' = \cos\phi(x - y\tan\phi)$

$Y' = \cos\phi(y + x\tan\phi)$

The multiplication by the tangent term can be avoided if the rotation angles and therefore tan (ϕ) 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 (ϕ) is represent able. With arc tan (2^{-i}) the cosine term could also be simplified and since cos (ϕ) = cos (-ϕ) it is a constant for a fixed number of iterations.

This iterative rotation can now be expressed as:

$X_{i+1} = k_i (x_i - y_i d_i 2^{-i})$

$Y_{i+1} = ki (y_i + x_i d_i 2^{-i})$

Where cos (arc tan (2^{-i})) and $d_i = \pm 1$. The product of the K_i's represents the so-called K factor: $k = \prod_{i=0}^{n-1} k_i$

This k factor can be calculated in advance and applied elsewhere in the system. A good way to implement the k factor is to initialize the iterative rotation with a vector of length k which compensates the gain inherent in the CORDIC algorithm. The resulting vector v' is the unit vector as

Those values arc tan (2^{-i}) can be stored in a small lookup table or hardwired depending on the way of implementation.

Since the decision is which direction to rotate instead of whether to rotate or not, d_i is sensitive to the sign of z_i therefore d_i can be described.

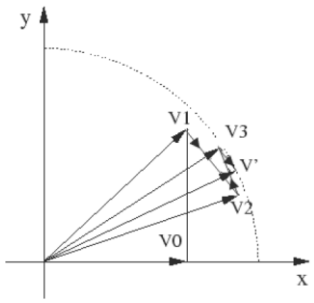

Fig. 2: Iterative vector rotations

Iterative vector rotations, initialized with V_0 simplified to the basic CORDIC-equations:

$X_{i+1} = [x_i - y_i d_i 2^{-i}]$

$Y_{i+1} = [y_i + x_i d_i 2^{-i}]$

The direction of each rotation is defined by d_i and the sequence of all d_i's determines the final vector. This yields to a third equation which acts like an angle accumulator and keeps track of the angle already rotated. Each vector v can be described by both the vector length and angle or by its coordinates x and y.

Following this incident, the CORDIC algorithm knows two ways of determining the direction of rotation: the rotation mode and the vectoring mode. Both methods initialize the angle accumulator with the desired angle z_0. The rotation mode, determines the right sequence as the angle accumulator approaches 0 while the vectoring mode minimizes the y component of the input vector.

The angle accumulator is defined by:

$Z_{i+1} = z_i - d_i \arctan(2^{-i})$

Where the sum of an infinite number of iterative rotation angles equals the input angle ϕ:

$$\phi = \sum_{i=0}^{\infty} d_i.arctan(2^{-i})$$

Those values arc tan (2^{-i}) can be stored in a small lookup table or hardwired depending on the way of implementation. Since the decision is which direction to rotate instead of whether to rotate or not, d_i is sensitive to the sign of z_i therefore d_i can be described as:

$$d_i = \begin{cases} -1: if & z_i < 0 \\ +1: if & z_i \geq 0 \end{cases}$$

With equation 3.19 the CORDIC algorithm in rotation mode is described completely. Note, that the CORDIC method as described performs rotations only within -π/2 and π/2. This limitation comes from the use of 2^0 for the tangent in the first iteration.

However, since a sine wave is symmetric from quadrant to quadrant, every sine value from 0 to 2π can be represented by reflecting and/or inverting the first quadrant appropriately.

Implementation of various cordic architectures

As intended by Volder, the CORDIC algorithm only performs shift and add operations and is therefore easy to implement and resource-friendly. However, when implementing the CORDIC algorithm one can choose between various design methodologies and must balance circuit complexity with respect to performance

A Bit-Serial Iterative CORDIC

Problems which involve repeated evaluation of a fixed set of nonlinear, algebraic equations appear frequently in scientific and engineering applications. Examples of such problems can be found in the robotics, engineering graphics, and signal processing areas.

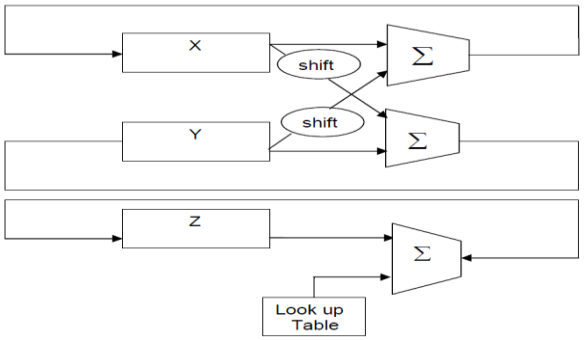

Fig. 3: transcendental functions

Evaluating complicated equation sets can be very time consuming in software, even when co-processors are used, especially when these equations contain a large number of nonlinear and transcendental functions as well as many multiplication and division operations. Both, the unrolled and the iterative bit-parallel designs, show disadvantages in terms of complexity and path delays going along with the large number of cross connections between single stages.

Original Bit-Serial Architecture

The original bit serial architecture of the bit-serial CORDIC architecture is shown. The figure shows architecture with 4 stages and a word length of 8-bits. As shown, the three paths the angular and two vector paths are different but highly dependent data paths.

The A/S units are one bit add/subtract units. In order to handle both addition and subtraction with the same hardware adder, there is an initial stage with a conditional inverter for one of the inputs for each A/S unit. To perform a subtraction the control bit of the adder, connected to the A/S unit is set to '1'. This inverts the second operand and sets the input sign bit to '1'. With this construct, no specific subtraction units are needed. The control bit is the sign bit of the previous stage angle operation. Thus angle calculation of the previous stage must be completed before the addition or subtraction can begin.

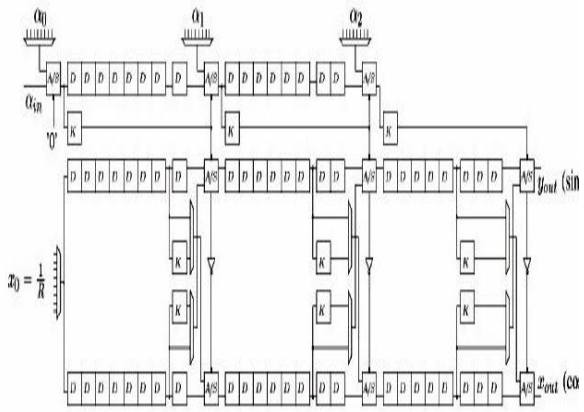

Fig. 4: CORDIC architecture

The constant angles, α_0 to α_{N-2} being the number of stages, are hard coded in the design. They are stored as negative values and fed to the A/S units of the angular path, bit by bit, through the MUXes. It is noted that the final stage in the CORDIC require no calculation in the angular path, since the values is not used in any later stage and is therefore not calculated.

Also in the x and y streams two integer bits, including the sign bit, are needed. The sign bit is needed since both x and y can take negative values (close to angles of 0 and π/2 respectively). The need for the extra integer bit is not obvious at first, but due to quantization, it is possible to over flow and end up at values of x, $y \geq 1$ in certain stages of the CORDIC. To safely handle this, an extra integer bit is needed. The original bit-serial architecture of the CORDIC is shown.

Proposed rchitecture

A major drawback of the presented bit-serial design is the many, long shift registers. These add to both the dynamic and static power dissipation of the circuit. Looking at the implementation, it is noted that many of the registers are present only to ensure that the vector paths are executed after the previous stage of the angular path.

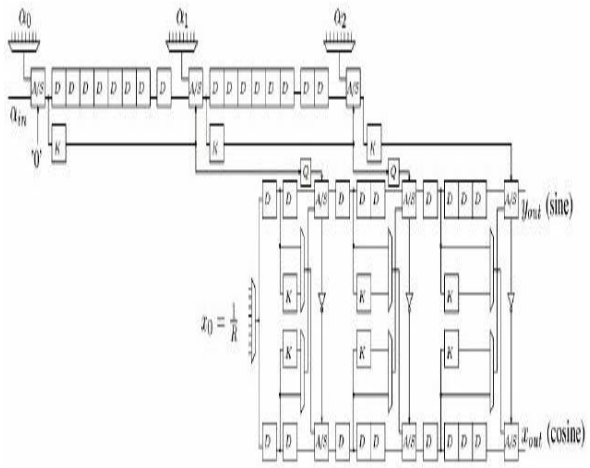

Fig. 5: improved architecture

Figure above shows this improved architecture. Removing the registers changes the timing of the circuit. In order for this to work, the angular path calculations must begin earlier than the corresponding vector path calculations. Also, to not increase the latency or reduce the throughput of the design, the vector path calculations should start so that the final add/subtract operation, , starts exactly when the final angular operation is finished.

Simulation Results:

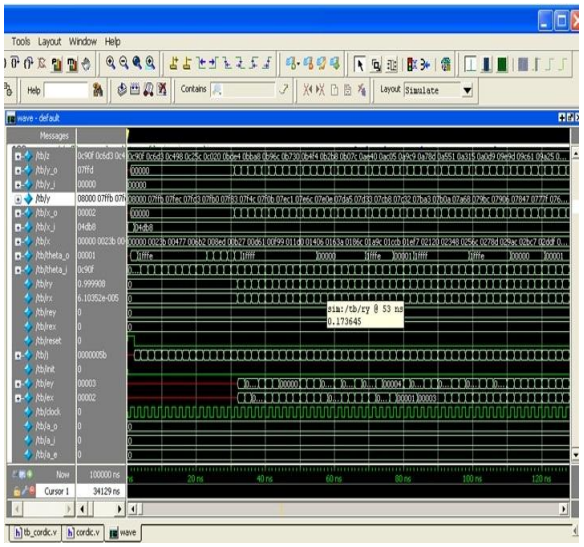

Fig. 6: Timing Diagram

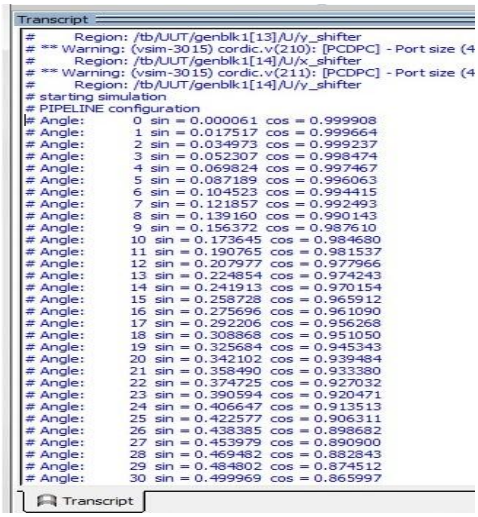

Fig. 7: Sine and cosine values from 0^o-30^o at Transcript window

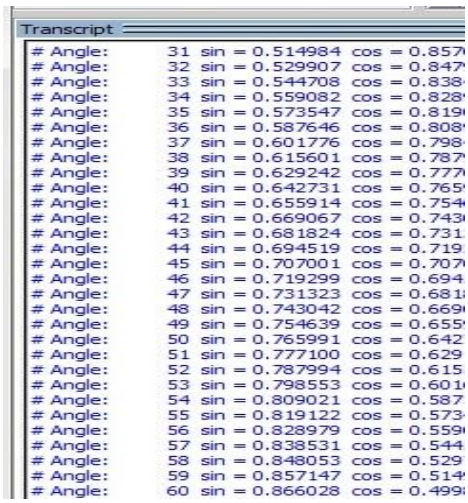

Fig. 8: Sine and cosine values from 31^o-60^o at Transcript window

Conclusion:

In this project, a new and improved bit-serial CORDIC architecture is presented. It is shown how it is possible to reduce the number of registers by calculating the angular path prior to the vector paths in the CORDIC. Some extra registers and extra logic are needed to store the sign bits of the angular calculations. However, this is substantially less than the area reduction in the vector paths. The improved architecture is 20 % smaller and consumes 26 % less power.

About Authors:

T. Chandrika is a P.G student of Department of ECE, Koushik college of Engineering, Visakapatnam, Andhra Pradesh, India. And D. P. Raju is presently working as Assistant Professor in the Department of ECE, Koushik college of Engineering, Visakapatnam, Andhra Pradesh, India.

References:

1. Johan Lofgren and Peter Nilsson, Dept. of Electrical and information technology, Lund University, Bit Serial CORDIC: Architecture and Implementation Improvements, Sweden
2. Volder J. E., .The CORDIC trigonometric computing technique., IRE Trans. Electronic Computing, Volume EC-8, pp 330 - 334, 1959.
3. Qian M., .Application of CORDIC Algorithm to Neural Networks VLSI Design., IMACS Multiconference on .Computational Engineering in Systems Applications (CESA)., Beijing, China, October 4-6, 2006.

4. Lin C. H. and Wu A. Y., .Mixed-Scaling-Rotation CORDIC (MSR-CORDIC) Algorithm and Architecture for High-Performance Vector Rotational DSP Applications., Volume 52, pp 2385- 2398, November 2005
5. Walther J.S.A, .Unified algorithm for elementary functions., Spring Joint Computer Conference, pp 379 - 385, Alantic city, 1971.
6. Kolk K. J. V., Deprettere E.F.A. and Lee A Floating Point Vectoring Algorithm Based on Fast Rotations., Journal of VLSI Signal Processing, Volume25, pp 125.139, Kluwer Academic Publishers, Netherlands, 2000.
7. Antelo E., LangT. and Bruguera J. D., .Very-High Radix CORDIC Rotation Based on Selection by Rounding., Journal of VLSI Signal Processing, Vol.25, 141.153, Kluwer Academic Publishers, Netherlands, 2000.