Implementation of Floating Point Cordic Coprocessor

Radhakrishnan K.R,
{krr.ece@psgtecg.ac.in}

Assistant Professor, Department of ECE, PSG College of Technology, Coimbatore.

Abstract. For real time applications like navigation, microprocessors are not fast enough but with the use of CORDIC, computation becomes faster. CORDIC is basically shift and add algorithm. The concept of a floating point The CORDIC coprocessor is designed to receive inputs in the IEEE 754 single precision format. Preprocessor converts these inputs into 32 bit fixed point representation. The CORDIC works for circular, linear, or hyperbolic systems in rotation or vectoring modes. A postprocessor converts the CORDIC module's output into the IEEE 754 format. Latency and area required for different CORDIC are analysed. According to the results, hyperbolic has the longest path delay and circular has the shortest, linear has the most LUT and register consumption, while Hyperbolic has the least. The use of shift-registers and adders instead of multipliers is the core premise of CORDIC, which saves a lot of hardware resources. Modified CORDIC with microrotations has less path delay when compared with the existing circular CORDIC. This increases the area and power values. Physical design is done in Cadence RTL Compiler and Encounter RTL to GDSII. CORDIC is utilised for polar to rectangular and rectangular to polar conversions, as well as the calculation of elementary functions and transformations such as the discrete Fourier transform (DFT) and the discrete cosine transform (DCT).

Keywords: Coordinate Rotation Digital Computer(CORDIC), Fixed point, Floating Point.

1 Introduction

In 1959, Jack E. Volder proposed the Cordic (COordinate Rotation Digital Computer) algorithm for the computation of trigonometric functions [1]. It is simple and iterative in nature. It is a cost-effective way to rotate vectors in the two-dimensional plane. Walther made extensions to the CORDIC algorithm that covers hyperbolic and transcendental functions [2].

A portion of Walther [2] is dedicated to the translation of fixed point to floating-point formats. The basic iteration equations are simply represented using floating-point arithmetic, as described earlier. Several quick remarks about changing the iteration sequence for floating-point are also available.

CORDIC allows you to work with rotations to compute sine, cosine, and arc tangent functions, as well as multiply and divide using simple shift and add steps. The CORDIC technique was not born out of need, with the incentive being the replacement of the B-58 bomber aircraft's analogue navigation computer with a digital computer.

The key challenge was determining current position on a spherical earth in real time. Since the HP35, CORDIC has been used in a variety of applications, including many pocket calculators and arithmetic coprocessors like the Intel 8085. Over time, not only has a wide range of CORDIC applications been proposed, but also much work has been made in the areas of

ICCAP 2021, December 07-08, Chennai, India
Copyright © 2021 EAI
DOI 10.4108/eai.7-12-2021.2314974

algorithm design and development of architectures for high-performance and low-cost hardware solutions [3]–[12]. Vector rotation has several uses in robotics, graphics, games, and animation [4].

Robot locomotion is accomplished through a series of short fixed-angle rotations. Fixed CORDIC rotations could be used to interpolate orientations between keyframes in computer graphics and animation. Uniform rotation can be seen in the movement of electrons within an atom, planets and satellites in the universe, and so on. In gaming, graphics, and animation, high-speed continuous rotation is required. Simulation, modelling, games, and animation all use objects with continuous rotations.

2 Proposed Design

A. PREPROCESSOR

The floating-point inputs are converted to fixed point representation by the preprocessor. The IEEE 754 single precision standardised format is used for the inputs. This 32-bit representation becomes a 32-bit fixed-point representation. The benefit of using floating-point representation is that it can accommodate a significantly wider range of numbers and is extremely accurate.

B. CORDIC

CORDIC is a collection of algorithms that shift and add data. The rotation mode and the vectoring mode are the two modes of operation for the CORDIC. A vector with initial coordinates (x0, y0) and a target rotation angle(z0) is given in the rotation mode operation, and the goal is to compute the final coordinate(x1,y1) by an iterative series of backward and forward rotations of the vector. The goal of the vectoring mode is to compute a vector's magnitude and phase angle given its initial and final coordinates.

Jack Volder [1] first created the CORDIC method for the circular coordinate system. It was later expanded to include linear and hyperbolic systems [2].

C. POSTPROCESSOR

The After CORDIC iterations, the result is 32 bits long. The results of the CORDIC are converted into 32 bit floating point representation by the postprocessor. It has been normalised.

The results are re-formatted into the IEEE754 single precision format for floating point numbers, which specifies the sign, exponent, and mantissa.

Jack Volder [1] first created the CORDIC method for the circular coordinate system. Walther discovered how to modify CORDIC iterations to compute hyperbolic functions [2] in 1971, and rebuilt the CORDIC algorithm into a generalised and unified version that can perform rotations in circular, hyperbolic, and linear coordinate systems. The new variable in the unified formulation is given distinct values for different coordinate systems. The CORDIC algorithm can be written as follows:

xi+1 = xi -µ$diyi$2 −i(1)

yi+1=yi + $dixi$2 −i(2)

zi+1=zi –$diei$

(3)

where μ=1 and ei= tan-1 (2-i)for circular system, μ=0 and ei=2-i for linear system and μ= -1 and ei=tanh-1 (2-i) for hyperbolic system[2].

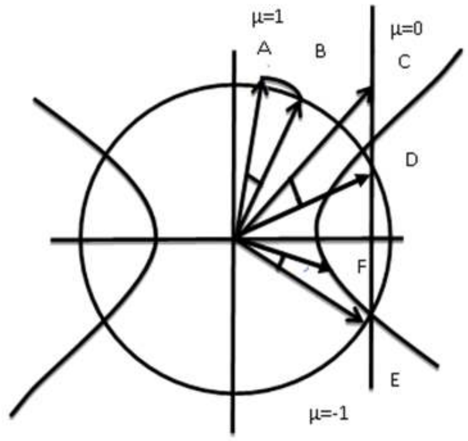

Fig1 Circular, linear and hyperbolic CORDIC

Figure 1 depicts circular, linear, and hyperbolic CORDIC. The sine and cosine can be computed using a circular CORDIC. Multiplication (rotation mode,y=0),multiply add,division (vectoring mode,z=0), and divide add may all be done with linear CORDIC rotations (vectoring mode).

In circular, computation of sine and cos can be done. simultaneously using rotation mode results

Xn = An [X0cosZ0– Y0sinZ0] (4)
Yn = A [Y0cos Z0+ X0 sin Z0] (5)
Zn=0 (6)

Setting X0=1/An is required to acquire sin and cos values. Then Yn Sin(Z0) and Xn Cos(Z0).

For linear rotations, results produced are
Xn = An [X0coshZ0– Y0sinhZ0] (10)
Yn = An [Y0cosh Z0+ X0 sin hZ0] (11)
Zn=0 (12)

To Setting Y0=0 and X0=1/An in the rotation mode is required to get these hyperbolic functions in X and Y. Then there is Yn Sinh(Z0) and Xn Cosh(Z0) (Z0).

The algorithm calculates the sine and cosine angle of a received vector by means of the arctangent function, and it is used to find the shift in phase of the input vectors in circular co-ordinate system. In the case of hyperbolic coordinate system it calculates sinh and cosh angle of a received vector. In linear co-ordinate system CORDIC processor calculate change in phase of x, y and z co-ordinates with respect to fixed angle of rotation.

.

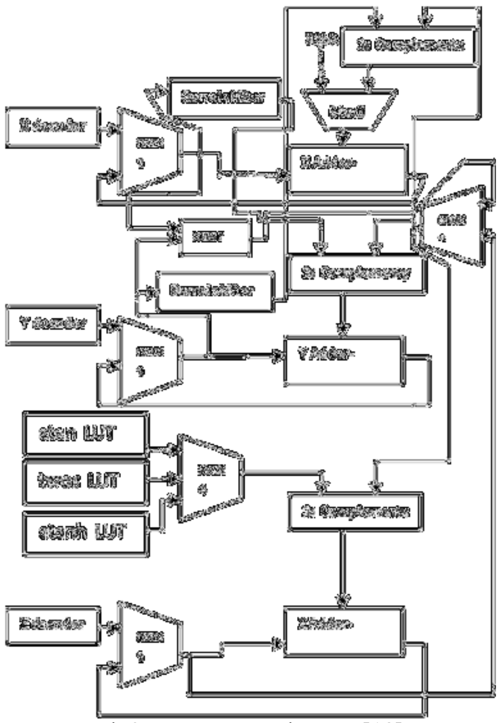

Fig2 CORDIC Arcitecture[18]

Micro-rotations can be utilised instead of fixed angle rotation for CORDIC. For the implementation of fixed-angle vector rotation, an optimised collection of micro-rotations is derived. It can be deduced from this that the bigger the number of micro-rotations, the higher the accuracy

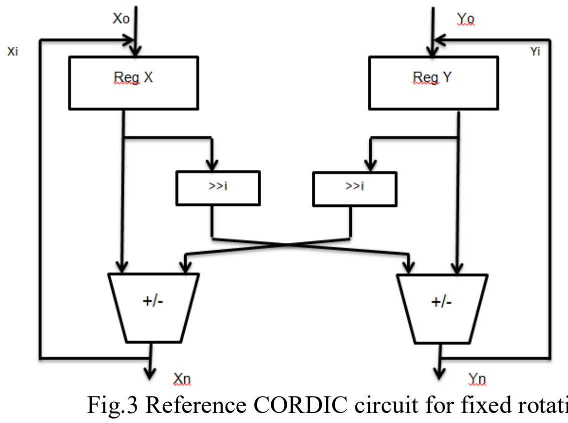

Fig.3 Reference CORDIC circuit for fixed rotation

It is shown here that using a rotation-mode CORDIC circuit to rotate a vector through a known and fixed angle of rotation, we can find a small number of predetermined elementary angles for I for 0 I m-1, where I = arctan(2-k(i)) is the elementary angle to be used for the micro-rotation in the CORDIC algorithm, and m is the minimum necessary number of micro-rotations. Meanwhile, it is generally known that a positive rotation through $0<Ø\leq \pi/4$ can be transferred into any angle $0< \theta \leq 2\pi$, without any additional mathematical operations [10]. The accuracy of the CORDIC algorithm is determined by how near the resultant rotation due to all micro-rotations approximates the desired rotation angle, which defines the deviation of the actual rotation vector from the estimated value.

Modified Circular CORDIC is developed which calculates sine and cosine values with more accuracy. This has micro-rotations instead of fixed angle of rotation. Full rotations are optimised with micro rotations, and the shifter is the component that consumes the most power and space. Reducing the number of components by half will give a 50% power reduction, by reducing the area. So modifying the CORDIC architecture can be done in order to reduce the power and area, if time delay is not considered.

3 Analysis And Results

In this paper, simulations were carried out and functionalities were verified using the ModelSim simulator. For this, Verilog HDL was used and synthesized using the Xilinx ISE Design Suite. Fig 3 shows the simulated sine and cosine values of certain angles calculated using 32-bit iterative CORDIC. Core has been simulated by operating it in the rotation mode for sine and cosine functions.

Table I provides latency of different CORDIC. Logic delay, Route delay, Total path delay are listed in the Table I. All the types are implemented with the same synthesis description.

.

TABLE I LATENCY OF DIFFERENT CORDIC

Parameter	CORDIC Architectures		
	Circular	***Linear***	***Hyperbolic***
Logic delay	6.364ns	8.315ns	10.790ns
Route delay	0.801ns	5.210ns	6.842ns
Total Path delay	7.165ns	13.525ns	17.632ns

Number of LUTs, Number of slices, Number of occupied slices and number of bonded IOS of Circular, Linear and Hyperbolic CORDIC are listed in the table II.

TABLE II AREA CONSUMPTION OF DIFFERENT CORDIC

Parameter	CORDIC Architectures		
	Circular	***Linear***	***Hyperbolic***
No of LUTs	1793	1950	1209
No of slices	928	1030	434

Parameter	**CORDIC Architectures**		
	Circul ar	***Linear***	***Hyperb olic***
No of occupied slices	935	1220	659
No of bonded IOS	9 7	65	52

TABLE III AREA CONSUMPTION OF MODIFIED CORDIC

Number of LUTs, Number of slices, Number of slice flipflops and number of bonded IOS of Modified Circular CORDIC are listed in Table III. Area of Modified Circular CORDIC is more than Circular CORDIC. There will be an increase in area due to micro-rotations.

Parameter	**CORDIC Architecture**
	Modified Circular
No of LUTs	1793
No of slices	928
No of occupied slices	935
No of bonded IOS	97

TABLE IV LATENCY COMPARISON OF DIFFERENT CORDIC

It is desirable to have less path delay in many applications. To reduce the path delay, Circular CORDIC is modified. Logic delay, Route delay, Total path delay are listed in the Table III. Modified CORDIC has less path delay compared to the existing Circular CORDIC.

Paramete r	**CORDIC Architectures**	
	Circular	***Modified Circular***
Logic delay	6.364ns	4.772ns
Route delay	0.801ns	2.129ns
Total Path delay	7.165ns	6.851ns

TABLE V AREA OF DIFFERENT CORDIC

Cadence RTL Compiler and Encounter RTL-to-GDSII system can be used for the physical design. Cadence RTL Compiler is used for RTL to Gate Level Netlist Synthesis.

Area report showing cells and cell area is generated. Number of Cells and cell area of Circular, Linear and Hyperbolic CORDIC are listed in the Table V. Linear has the least number of cells and cell area.

Table VI also shows the static power dissipation values obtained for Circular, Linear and Hyperbolic CORDIC from Encounter RTL-to-GDSII system.

Variants of CORDIC	**Power Dissipation**			
	Leakage Power (mW)	Internal Power (mW)	Switching Power (mW)	Total Power (mW)
Circular	0.06221	3.483	1.454	4.99
Linear	0.01453	0.2475	0.1141	0.3761
Hyperbolic	0.06806	5.236	4.365	9.669

TABLE VI POWER DISSIPATION VALUES OF DIFFERENT CORDIC

Results shows that Linear CORDIC has the least power dissipation value whereas hyperbolic CORDIC has the higher power dissipation value.

The CORDIC Coprocessor architecture, which includes Linear, Hyperbolic, and Circular functions, is well suited to solving trigonometric relations at high speeds. It can be utilised in real-time applications, but the main benefit is the ability to generate multiple elementary functions from a single unit.

The algorithm requires adders, multipliers, and shift registers. Shift operations simplifies multiplication. The ATR constants are distributed separately to each adder, allowing for a hardwire approach rather than the use of a specialised ROM. Look Up Tables are used to implement the angular values for the hyperbolic and circular functions (LUT).

The shifters do not need to be programmable, which is a beneficial advantage. Nonetheless, a trade-off between area and accuracy must be made such that the design's usefulness takes into account the precision bits required for each arithmetic operation as well as the hardware resources available.

4. Conclusion

This work introduced a multi CORDIC architecture that combines multiple CORDIC architectures and allows the user to choose the CORDIC architecture that best meets his needs. When compared to previous multi CORDIC architectures, an area reduction of up to 10% is achieved. It was broken down into three sections: preprocessor, CORDIC calculations, and postprocessor. According to synthesis, hyperbolic has the longest path latency while circular has the shortest. Linear has the most LUT and register consumption, while Hyperbolic has the least.Modified CORDIC with microrotations Instead of fixed angle rotations has less path delay when compared with the existing circular CORDIC. Physical design has been done in Cadence RTL Compiler and Encounter RTL to GDSII. Area and Power values are obtained. Linear shows the least area and power whereas Circular has the highest. Layouts are obtained for Circular, Linear and hyperbolic CORDIC. Floating point CORDIC processor mentioned in this paper can be used for guidance and control of launch vehicle and satellites.

References

[1] M. Tholkapiyan, A.Mohan, Vijayan.D.S , "A survey of recent studieson chlorophyll variation in Indian coastal waters", IOP Conf. Series: Materials Science and Engineering 993 (2020) 012041, doi:10.1088/1757-899X/993/1/012041.

[2] Kiruthika, C., S. Lavanya Prabha, and M. Neelamegam. "Different aspects of polyester polymer concrete for sustainable construction." Materials Today: Proceedings 43 (2021): 1622-1625.

[3] Prabha, S. Lavanya, M. Surendar, and M. Neelamegam. "Experimental investigation of eco-friendly mortar using industrial wastes." Journal of Green Engineering 9.4 (2019): 626-637.

[4] C. Amuthadevi, D. S. Vijayan, Varatharajan Ramachandran, "Development of air quality monitoring (AQM) models using different machine learning approaches", Journal of Ambient Intelligence and Humanized Computing, https://doi.org/10.1007/s12652-020-02724-2

[5] Gayathri Monicka,J, Dwarakesh,C. Amutha Devi, "Renewable based Multicarrier PWM Topology for Symmetric MLI" International Journal of Engineering and Technology, Vol 8 No 6 Dec 2016-Jan2017,PP 2902- 2911

[6] Amutha devi, Gayathri Monicka.J, "Emerging bio-medical applications and open research challenges in cognitive internet of things (CIOT)" International Journal of Pharmacy and Technology" Dec-2016 Vol. 8 Issue No.4 5049- 5054.