ISSN: 2582 - 6379
Orange Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2024, June, Volume 6 Issue 1
www.ijisea.org

Design and Implementation of Trigonometric Functions with High Speed CORDIC Algorithm

Bonamsetty Jahnavi[1], K.Chandra Sekhar[2]
PG Scholar[1], Assoc. Professor[2]
Department of Electronics and Communication Engineering
Sri Sai Institute of Technology and Science
Masapeta,Rayachoty, Andhra Pradesh, India

ABSTRACT

The Present work focuses on realization of the exponential and converse exponential functions by utilizing the CORDIC algorithm. The exponential function finds its application in the areas of signal processing, image processing and video processing. The realization of the exponential functions forms the bottleneck for refining the performance of the design in terms of area and delay. The range of the inputs are extended from -7 to 7. The inputs are scaled by 65536 to improve the accuracy of the result. The output values are 1 floating point values and hence they are scaled by the 65536 to get the actual values. The exponential function was realized using the CORDIC algorithm in which addition/subtraction and shift operations were involved and hence the performance can be improved. The throughput of the model was enhanced by the utilizing the pipelining of the design. The algorithm has been realized using vhdl in the stratix II FPGA and the algorithm was synthesized using the Quartus -11 9.1 SP2 software.

Keywords: Pipelined, Parallel,CORDIC, arcsine, arccosine, sine, cosine, FPGA, ASIC

I.INTRODUCTION

The advancement of very large-scale integrated circuit (VLSI) technology and the emergence of ZADS electronic design automation (EDA) tools have led to current research in digital layout (DSP), communications, and other functions. Fast VLSI architecture. Gordon Moore predicted that the cost of VLSI technology would increase, and since 1965 the industry has developed a new technology that fits his prediction curve, called Moore's Law. These advances give designers the power to transform algorithms into architectures. Many DSP algorithms use simple operations such as logarithm, trigonometric, exponential, division and multiplication. Two ways to accomplish this task are using lookup tables and polynomial expansion. The above operation requires a lot of multiplication/division and addition/subtraction. Logarithms, exponents, hyperbolas, multiplication, division, square roots, etc. Its simplicity is that one can calculate any of the above functions using transformation and addition of the form x±2-i*y. Many signal processing and communications systems operate CORDIC in a rotational or vector format in a coordinated environment. The COordinate Rotation DIgital Computer (CORDIC) algorithm was first introduced by J E Volder in 1959 [1], generalized and unified by Walther in 1971 [2] and Later-on a lot of work has been done. The unified CORDIC algorithm utilizes the coordinate rotations to evaluate a lot of computational complex mathematical functions such as trigonometric, hyperbolic, exponential, logarithmic functions, multiplication, division and square root. CORDIC is an attractive algorithm because of its unique property of using basic operations of the form (a ± b >> 2 ~) using simple hardware to compute most of the mathematical functions.

At the time of invention of the CORDIC, the multipliers and divider were very expensive in hardware. The implementation of CORDIC algorithm requires fewer complex hardware than the conventional method and is particularly well-suited for applications in which cost (chip gate count has to be minimized) is much more important than speed. Therefore, CORDIC was an attractive choice for computing elementary functions instead of using various polynomial methods. CORDIC made its first hardware appearance in HP's Pocket

ISSN: 2582 - 6379
Orange Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2024, June, Volume 6 Issue 1
www.ijisea.org

calculator HP 35, the arithmetic Co-processor of Intel 8087 [3], RADAR signal processors and Robotics[4]. The CORDIC has also been proposed for computing Discrete Fourier and Discrete Cosine Transforms [4], Discrete Chirp-Z and Hartley [5], Transforms, Filtering [4], Singular Value Decomposition [6] and Solving Linear System of Equations [7]. Fig.1 shows the basic CORDIC diagram.

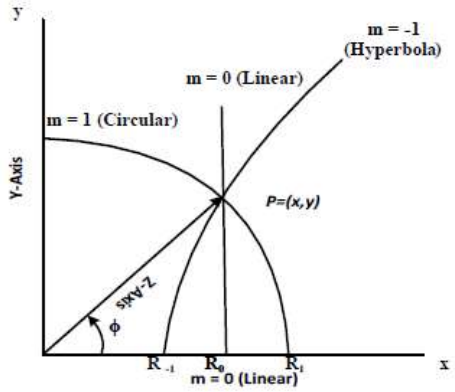

Fig.1. Basic CORDIC

Originally, CORDIC was programmed to solve either set of the following equations:

$$x' = K(x\cos\theta - y\sin\theta) \quad (1)$$

$$y' = K(y\cos\theta + x\sin\theta) \quad (2)$$

Or

$$R = K\sqrt{x^2 + y^2} \quad (3)$$

$$\theta = tan^{-1}\frac{y}{x} \quad (4)$$

where k is a constant.

The first set of equations is solved in a CORDIC operation called Rotation operation mode (RM) and the second set is solved in Vectoring operation mode (VM). In rotation operation, the coordinate components of a two-dimensional vector and a rotation angle are given; and the coordinate components of the vector after rotation are computed. In vectoring operation, the coordinate components of a vector axes are given; and the magnitude and the angle of the original vector with the x-axis are computed. This brief presents a novel CORDIC-based approach for arcsine and arcosine computation that solves the issues with the direct CORDIC-based approach without increasing the complexity of the stages. The proposed approach is based on the approximation of the gain of the microrotations by an expression that can be computed only with bit-wise shifts and additions. This approach leads to accurate computations of the arcsine and arccosine functions using fewer resources than previous approaches. Additionally, an FPGA implementation is provided and compared to state-of-the-art CORDIC-based implementations for arcsine and arcosine computations.

2. Background:

The main issue with the CORDIC-based arcsine and arccosine algorithms is the compensation of the gain over *Ti* at each iteration. Avoiding this computation leads to large approximation errors at several points across the domain of the functions, as reviewed in Section II-B. Using a real multiplier to compensate this gain is often too expensive in terms of area for hardware implementations. The double iteration CORDIC, reviewed in Section II-C solves this issue without requiring real multipliers. However, the complexity of the stages is increased. This leads to hardware implementations that require more resources.

In this brief, we propose a method to compute *Ti* using only additions and bit-wise shifts. This approach does not require to modify the *X*, *Y* and *Z* paths of the original CORDIC. This leads to a simple, low-area circuit capable of computing accurately the arcsine and arcosine functions.

ISSN: 2582 - 6379
Orange Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2024, June, Volume 6 Issue 1
www.ijisea.org

it can be observed that for $i \geq 0$ the term $2-4i-2$ is considerably smaller than the other two terms. Thus, it can be neglected without leading to large errors in the CORDIC computation.

By applying this idea, the value of T in the proposed approach is calculated as $Ti+1 = Ti + Ti \cdot 2-2i-1$.
By following the ideas in [10], the proposed algorithm can be further optimized by avoiding the first iteration. This is due to the fact that the arcsine and arccosine functions are odd and only have outputs in the range $[-\pi/2, \pi/2]$ and $[0, \pi]$, respectively. For both of them, the input is in $[-1, 1]$, but we can consider only the absolute value of the input. This will provide a solution in $[0, \pi/2]$. Then, if the input was in the range $[0, 1]$, the result is correct, whereas if the input was in $[-1, 0)$, the output is moved to the fourth quadrant for arcsine by negating the output, and to the second quadrant for arccosine by subtracting the result from π.

3. Proposed Architecture

3.1.Parallel CORDIC

In this type of architecture, all the iterations take place in a single clock cycle and shown in Fig.1. In this the performance will work in parallel. The time consumption will take less ,because all arctan will perform in parallel

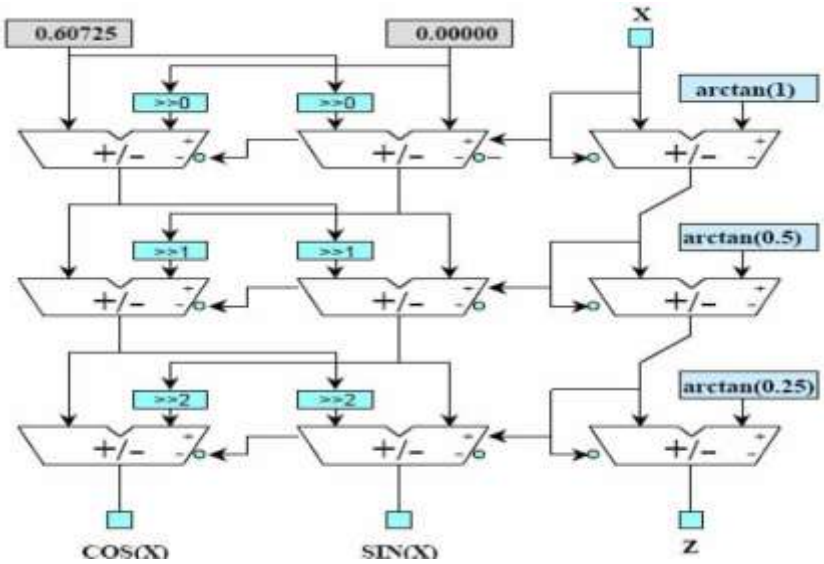

Fig.1. Parallel CORDIC architecture

3.2. Pipelined CORDIC Architecture

It is comparatively the most efficient CORDIC architecture. In this method multiple iterations take place in multiple clock cycles. It is implemented by inserting registers within the different adder stages. The architecture is shown in Fig.2. Depending upon the application, CORDIC Processer is applied in number of ways. The simple structure is sequential structure involve three adder/subtractor and two shifter with a rom containing search table. Serial structure performs one small spinning for every time pattern. Outcome is acquired after n time pattern. Since sequential structure uses n time pattern for every spinning hence it is very slowly. Fig.3 reveals the sequential architecture.

ISSN: 2582 - 6379
Orange Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2024, June, Volume 6 Issue 1
www.ijisea.org

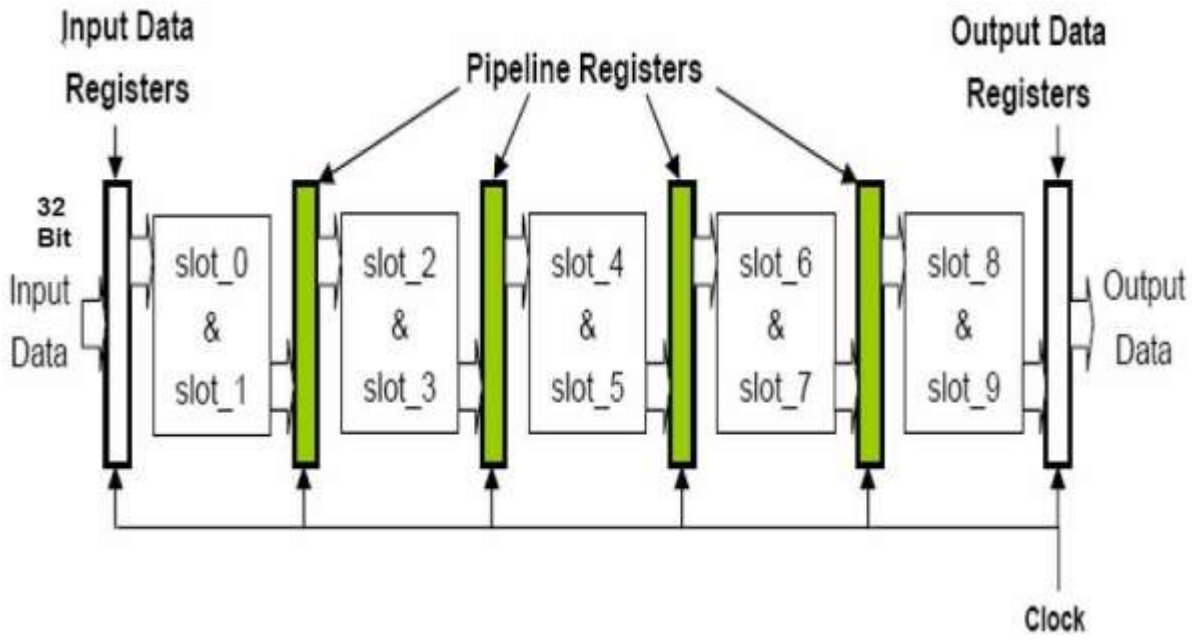

Fig.2 Block diagram of Pipe-lined CORDIC Architecture

.

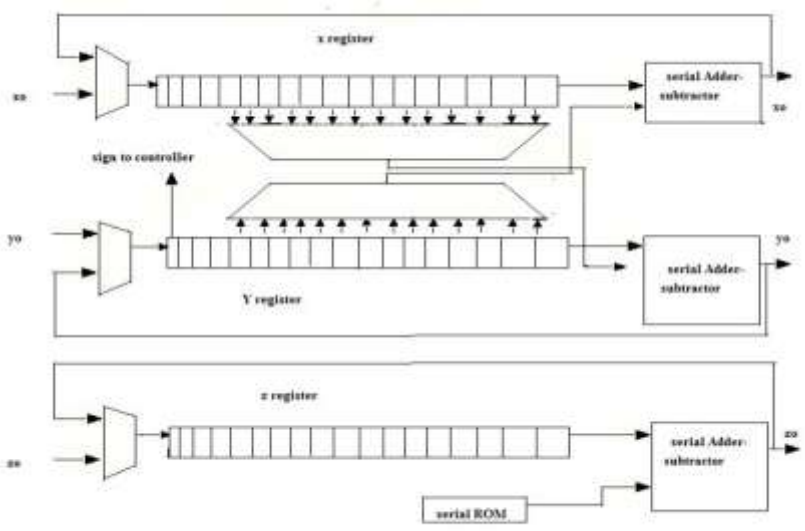

Figure.3 Serial architecture

The pipeline converts patterns into instructions. There are n number of stages Section Suspension. The first result of n-level CORDIC is obtained after n samples. Afterwards results were obtained after each sample. Pipeline model with mobile subscription.

ISSN: 2582 - 6379
Orange Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2024, June, Volume 6 Issue 1
www.ijisea.org

4. Comparison and Experimental results

Table I compares the number of adders, subtracters and registers required to implement a stage for the different CORDIC-based arcsine approaches. The direct CORDIC approach requires the smallest area with 3 adders and 3 subtractors, but it has high approximation error. Conversely, both the double iteration CORDIC and the proposed approach obtain accurate results for the whole domain of the arcsine and arccosine functions. Among them, the proposed approach saves 33% of the adders/subtracters, as it only requires 4 of them, whereas the double iteration CORDIC requires 6 of them. The proposed architecture has been described in VHDL. Table I shows experimental results of the proposed approach and previous CORDIC-based pipelined arcsine implementations. The proposed implementations A and B have been provided with the only purpose of comparing to previous works in a fair way, whereas implementation C provides results for a state-of-the-art FPGA. In all the proposed implementations, the error has been calculated as the maximum angular error among 106 uniformly distributed inputs in the range [−1, 1]. Implementation A and [11] use the same setup with 25-bit word length, 12 iterations and an Altera Cyclone EP4CE115F29C7 FPGA. It can be observed that both obtain similar throughput and error. However, the proposed implementation A reduces the latency by 10% and requires significantly less area, with savings of 60% in terms of LUTs and reduction of the number of FFs by a factor 4. Implementation B and [10] have 20-bit word length, 12iterations, and use a Xilinx Virtex-6 XC6VLX240T FPGA. Compared to [10], the proposed approach B reduces the number of LUTs by 20% and has a similar number of FFs. This agrees with the information provided in Table I. Furthermore, the proposed approach B achieves 20% higher throughput, which results in lower latency in nanoseconds, even when the implementation in [10] takes one less clock cycle to process the inputs. Finally, the error of the proposed approach is 7% higher than that in [10], but it could be reduced by increasing the number of iterations by one at the cost of a slight increase in area. Finally, the proposed implementation C considers the same design as B, implemented on a Xilinx Virtex Ultrascale+ XCVU37P FPGA. On this FPGA, the design reaches a throughput of 595 MS/s with slightly less LUTs, same number of FFs and much better ratio between throughput and power consumption than implementation B on a Virtex-6.

	Adders/Substractor	**Registers**
Direct CORDIC	**3**	**3**
Proposed design	**4**	**4**

Table.I Comparison of the resources required to implement a stage for an Arcsine based CORDIC

Conclusion:

In this brief, a novel algorithm based on CORDIC for the computation of arcsine and arccosine functions has been presented. Contrary to previous CORDIC-based approaches, the proposed algorithm does not require neither to increase the complexity of the iterations nor to compute real multiplications in order to compute the functions accurately. The hardware implementation of the proposed algorithm has been provided, proving that a better trade-off between area and error is obtained when compared to other CORDIC-based implementation.

REFERENCES:

[1] C. Mazenc, X. Merrheim, and J.-M. Muller, “Computing functions cos/sup -1/ and sin/sup -1/ using CORDIC,” *IEEE Trans. Comput.*, vol. 42, no. 1, pp. 118–122, Jan. 1993.
[2] X. Liu, Y. Xie, H. Chen, and B. Li, “Implementation on FPGA for CORDIC-based computation of arcsine and arccosine,” in *Proc. IET Int. Radar Conf.*, Oct. 2015, pp. 1–4.
[3] W. Zhu, S. Ye, Y. Huang, and Z. Xue, “Design of a precise subdivision system for gratings using a modified CORDIC algorithm,” *IET Circuits Devices Syst.*, vol. 13, no. 8, pp. 1284–1291, Nov. 2019.
[4] A. Saha, A. Ghosh, and K. Kumar, “FPGA implementation of arcsine function using CORDIC algorithm,” *Adv. Model. Anal.*, vol. 54, no. 2, pp. 196–201, Sep. 2017.
[5] T. Lang and E. Antelo, “CORDIC-based computation of arccos and arcsin,” in *Proc. IEEE Int. Conf. Appl. Specific Syst. Architect. Process.*, Jul. 1997, pp. 132–143.

[6] M. Zechmeister, “Solving Kepler’s equation with CORDIC double iterations,” *Monthly Notices Roy. Astron. Soc.*, vol. 500, no. 1, pp. 109–117,Jan. 2021.
[7] W. Zhu, S. Ye, Y. Huang, and Z. Xue, “An improved CORDIC for digital subdivision of Moiré signal,” *Metrol. Meas. Syst.*, vol. 27, no. 1, pp. 51–64, Mar. 2020.
[8] F. Salehi, E. Farshidi, and H. Kaabi, “Novel design for a low-latency CORDIC algorithm for sine-cosine computation and its implementation on FPGA,” *Microprocess. Microsyst.*, vol. 77, pp. 1–7, Sep. 2020.
[9] J. P. Lim, M. Shachnai, and S. Nagarakatte, “Approximating trigonometric functions for posits using the CORDIC method,” in *Proc. Int.2022.*