AREA-EFFICIENT CORDIC ARCHITECTURE WITH LUT ASSISTANCE FOR FIXED-POINT - ARCTANGENT COMPUTATION

RAJESH DURGAM
Assistant Professor in ECE
rajeshdurgam2001@gmail.com
Malla Reddy College of Engineering

Dr.P.SUDHAKARA REDDY
Professor in ECE
drpsr2022@gmail.com
Malla Reddy College of Engineering

ABSTRACT
Because it uses straightforward shift-and-add operations to carry out mathematical operations without the need for sophisticated multiplier hardware, the Coordinate Rotation Digital Computer (CORDIC) technique is frequently used for fixed-point arctangent computing. However, in order to get the required angular accuracy, traditional iterative CORDIC systems need several rounds, which might lengthen calculation times and use more hardware resources. An area-efficient Lookup Table (LUT) is proposed in this study.CORDIC architecture with assistance for computing fixed-point arctangents. In order to give an initial angular estimation and minimise the amount of iterative operations needed for final angle computation, the suggested design combines a LUT-based coarse-angle estimate technique with the traditional CORDIC algorithm. Input processing, quadrant identification, LUT-based angle estimates, optimised CORDIC iterations, and angular correction stages are all included in the architecture. The design maintains fixed-point arithmetic to facilitate effective implementation of digital electronics. Verilog HDL is used to describe the design, and Xilinx Vivado 2026 is used for functional simulations and synthesis.1. Hardware, area, power consumption, calculation delay, and angular precision are used to evaluate the performance. The suggested method seeks to minimise the hardware complexity of traditional fixed-point CORDIC implementations while striking a compromise between computing efficiency and numerical precision.
Keywords: Verilog HDL, area optimisation, FPGA, look-up table (LUT), arctangent computing, CORDIC, fixed-point arithmetic, and angle estimation
.

I.INTRODUCTION

Digital technology frequently uses fixed-point arithmetic due to its predictable word length, effective storage, and comparatively straightforward arithmetic circuits. Standard trigonometric functions as well as coordinate transformations must be computed for many signal processing and control applications. Among these, phase prediction, digital interaction, robotics, instrumentation, mechanical control, and image processing systems all depend on the arctangent. The two-input arctangent calculating function, atan2(y,x), calculates a vector's angle while taking into account the signs for both input coordinates.

The well-known shift-and-add algorithm CORDIC is used to evaluate linear, hyperbolic, and trigonometric functions. Volder first used it to compute trigonometric functions in real time, and Walther later extended it to a larger family of fundamental functions [1], [2]. When in vectoring state, CORDIC accumulates the angle of rotations while repeatedly driving the vertical position toward zero. The ultimate collected angle is the input vector's arctangent. A series of micro-rotations are carried out using a standard radix-2 CORDIC device. In the first iteration, the direction is chosen according to the residual Y coordinate's sign. A precalculated arctangent constant is used to update the Z value, while shifted coordinates are added or subtracted to update the X and Y values. Even if the processes are straightforward, numerous iterations could be necessary to achieve a high-accuracy output. A sequential architecture has a lengthy calculation latency since these iterations are carried out one after the other. An unrolled architecture may require more hardware space since it instantiates several phases.

Several methods, such as angle recoding, radix-4 processing, a redundant arithmetic, precomputation,. iteration skipping, and pipelining, have been studied to enhance CORDIC performance [3]–[10]. When a coarse geometric region can be identified with a limited amount of input bits, LUT-based approaches become more appealing. While the reduced CORDIC engine calculates the remaining angular error, a stored table can be used to generate a coarse estimate. If the LUT precision and remaining angle range are carefully chosen, this hybrid technique can minimise the quantity of fine revolutions needed.

In this paper, an area-efficient CORDIC structure for fixed-point tangent arc computation with LUT assistance is presented. Prior to being examined for sign, the supplied I and Q numbers are normalised. The input is converted via quadrant identification and mapping into a region that the core algorithm can support. An optimised CORDIC iteration unit calculates the residual angle, while a LUT offers a coarse angle estimate. The final static-point

arctangent output is obtained by combining the coarse & residual angles, then performing angle rectification and quadrant restoration.

The primary contribution is a fixed-point datapath that combines reduced-iteration CORDIC refinement with coarse-angle LUT estimation. The goal of the architecture is to maintain CORDIC's shift-and-add functionality while reducing the number of repetitive micro-rotation operations. Verilog HDL is used to describe the design, and Xilinx Vivado 2026.1 can use simulation, synthesis, and timing analysis to assess it.

II. RELATED WORK

The original algorithm for CORDIC [1]. Volder developed CORDIC, a technique that uses binary shifts, iterative additions, and subtractions to compute trigonometric functions. The vectoring and rotational operations that serve as the foundation for numerous hardware arctangent units were established in this study.

CORDIC in general [2]. Walther added support for circular, a linear, oblique, and hyperbolic coordinate structures to the original CORDIC formulation. The unified formulation offered a standard mathematical structure to support hardware implementations and expanded the use of CORDIC to basic functions. VLSI Architectures Based on CORDIC [3]. Hu talked on digital signal processing implementation options and examined CORDIC-based VLSI architectures. The study emphasises how crucial architecture choice is for striking a balance between computing precision, area, and throughput.

CORDIC Implementations Based on FPGAs [4]. Andraka reviewed CORDIC methods for computers with FPGAs. In addition to discussing useful architecture concerns for FPGA implementation, the study illustrates why shift-and-add arithmetic is appropriate for programmable logic. Survey of CORDIC Architecture [5]. CORDIC designs were categorised by Lakshmi and Dhar based on rotation mechanism and number representation. Their survey highlights the trade-off between hardware replication and sequential reliance while describing parallel and low-latency techniques.

CORDIC Research for 50 Years [6]. The evolution of CORDIC algorithms, underlying architectures, and applications was examined by Meher et al. The evaluation offers a comprehensive basis for choosing an optimised CORDIC architecture and addresses accuracy, speed, area, and power factors.

Arctangent CORDIC Modified [7]. A modified arc tangent CORDIC approach with variable iterations was introduced by Zhu, Ge, and Jiang. In contrast to a predefined iteration sequence, their technique chooses appropriate rotation angles to enhance convergence and decrease the number of iterations.

Optimisation of atan2 Based on LUT [8]. An optimised CORDIC-based atan2 solution for FPGA devices was presented by Torres, Valls, and Canet. Their approach reduces the area of the unrolled design without sacrificing performance by using FPGA LUT capabilities for the Z-path. The LUT-assisted angle computing taken into consideration here is closely related to this study. Angle-Recoding CORDIC and LUT [9]. Research on angle recoding and table-lookup rotation integrates redundant-rotation detection with precomputed rotation information. These methods seek to lower the rotation datapath's complexity and the number of occupied rounds. Quantisation and Fixed-Point Accuracy [10], [11]. Word length, rounding, truncation, angle-constant precision, and iteration count all affect fixed-point CORDIC accuracy. Numerical error must be assessed in conjunction with architectural and input magnitude, according to research on CORDIC quantisation and real-time arctangent computing.

High-Radix CORDIC and Radix-4 [12], [13]. Larger elementary rotations are used in high-radix CORDIC techniques to minimise the number of phases. Although these techniques can reduce latency, they might need precomputed constants and more intricate selection logic. Instead, the suggested LUT-assisted method employs a decreased fine-rotation sequence after a coarse initial estimate.

CORDIC pipelined architectures [14], [15]. In order to increase clock frequency and enable the overlapping processing of many vectors, pipelined CORDIC architectures place registers in between processing stages. Increased registers and clock dispersion are their primary costs, which need to be taken into account in an area-efficient design.

Parallel and Redundant CORDIC [16], [17]. To lessen the dependence between subsequent CORDIC iterations, redundant arithmetic and simultaneous direction choice have been suggested. These techniques encourage the use of precalculation in the suggested architecture and offer substitutes for traditional sequential vectoring.

III. PROPOSED SYSTEM

The suggested architecture combines a decreased-iteration CORDIC vectoring unit with a LUT-based coarse-angle estimator. Angle combination, final angle correction, optimised CORDIC iteration stages, LUT-assisted coarse-angle estimates, quadrant recognition and mapping, and input processing make up the entire datapath. The suggested block diagram shows that the result is an 18-bit fix-point arctangent value.

A. Input Processing: I_p and Q_p are signed 17-bit numbers that are used as input components. In order to ensure that the external datapath functions within a specified range, input processing verifies the signs, manages the most important sign bits, and normalises the magnitudes. Normalisation prevents needless loss of accuracy for small values and helps preserve numerical performance across input magnitudes. Since atan2(0,0) is undefined, the zero-vector situation needs to be treated independently.

B. Quadrant Mapping and Detection: The input quadrant is identified by the symbols of I_p and Q_p. Before the core estimation, the input vector is transferred to a primary

quadrant or other supported region. When restoring the final angle, the quadrant data is kept as metadata. With this method, the LUT and decreased CORDIC unit can function within a limited angular range.

C. LUT-Based Coarse-Angle Estimation: A LUT address is formed using the normalised input bits. A crude angle estimation, θ0, or matching angle anchor is stored in each LUT entry. The input phase can be roughly estimated from the LUT output. The coarse angular precision and memory needed depend on the number of addressing bits and the depth of the table. Although it adds storage and selection logic, a larger table can decrease the residual angle.
D. Optimised CORDIC Iteration Unit: The residual angled is refined using fewer CORDIC iterations following coarse-angle estimate. The residual vector is used by the vectoring unit to update coordinates via shift-and-add. Fixed-point values are kept for the iteration constants. The needed output quality and the LUT's resolution are taken into consideration when choosing the quantity of fine iterations.
E. Coarse and Residual Angle Combination: To get the estimated angle θest, the coarse estimate θ0 and leftover angle θr are combined. A consistently signed fixed-point encoding is used for the addition. Fractional bits must be preserved during the addition process and overflow must be avoided.

F. Alignment Correction and Sector Restoration: Using the original quadrant data as a basis, the final adjustment block applies the necessary offset or sign change. The output is structured as an 18-bit arctangent. The RTL and testbench must have uniform specifications for the angle range and coding standard, such as signs radians or a normalised phase representation.

G. Fixed-Point Display: The provided architecture illustrates that the input size is 17 bits and its output size is 18 bits. Selecting standard internal X, Y, and Z lengths is necessary to protect guard bits during adds and shifts. The same scaling approach must be applied when quantising the LUT and arctangent constants. When hardware costs allow, rounding should be chosen over unregulated truncate at the final output.

H. Verilog HDL Module Organization: The layout can be divided into a top-level arctan_lut_cordic module, angle_combiner, quadrant_corrector, coarse_angle_lut, cordic_residual_engine, input_normalizer, and quadrant_mapper. Input and output widths, clock/reset parameters, and valid-signal behaviour should all be precisely specified for each submodule. This modular design makes debugging and simulation easier.
I. Verification Strategy: For atan2 (Q,I), the testbench ought to contrast the RTL outputs with a high-precision program or testbench reference. All four quadrants, positive and negative axes, values close to zero, maximum negative and negative inputs, and angles close to quadrant boundaries should all be included in test vectors. Additionally, the testbench should verify output-valid timing, reset operation, and the zero-vector convention. By supplying a coarse initial angles from the LUT, the suggested architecture aims to minimise the quantity of fine CORDIC iterations. LUT size, number of remaining iterations, fixed-point precision, synthesising mapping, and destination FPGA device all affect the real area and timing benefit. As a result, the architecture should be assessed using the same test vectors and limitations as the conventional and suggested designs.

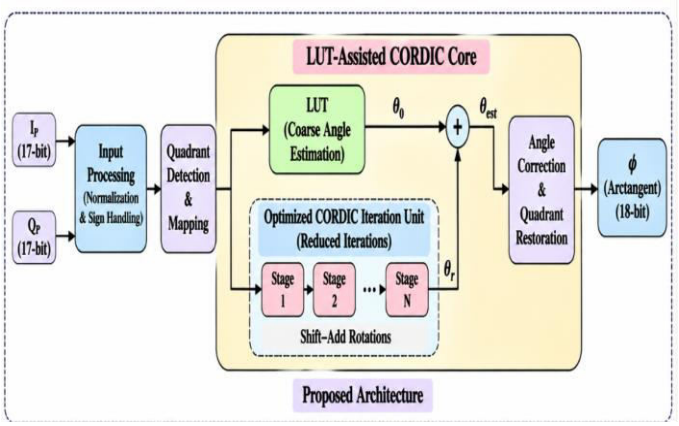

Fig. 1. Proposed LUT-assisted CORDIC architecture for fixed-point arctangent computation.

IV. RESULTS AND DISCUSSION

To confirm that the LUT-assisted design generates an arctangent arc estimate for sign input vectors, functional simulation is employed. The testbench should verify that the quarter restoration algorithm generates the right output range by examining every input quadrant. Additionally, it should confirm the residual CORDIC completion and make sure the proper fixed-point scaling is applied when combining the coarse and residue angle data. Because the LUT offers an initial coarse-angle estimate, the suggested architecture is anticipated to reduce the amount of fine CORDIC repeats. The active calculation sequence can be shortened or the quantity of recurrent coordinate-update logic can be decreased with fewer residual iterations. The final area improvement, however, is dependent on the CORDIC structure and table size because the LUT adds storage and address-selection logic. In terms of hardware utilisation, power, time, processing cycles, and angular precision, Table I contrasts the suggested LUT-assisted CORDIC design with the traditional fixed-point CORDIC architecture.

The resource and temporal features of the two designs are displayed in the comparison. While the LUT offers a coarse initial angle that lowers the residual computation needed by the CORDIC core, the LUT-assisted structure employs fewer fine CORDIC iterations. The target FPGA device, synthesis mapping, fixed-point precision, and LUT depth all affect the total trade-off.

The absolute error of the fix-point RTL outputs and a highly precise atan2(Q,I) reference should be used to assess angular correctness. The root-mean-square error, mean absolute error,, and maximum absolute error should all be expressed in radians or degrees. Both designs must have identical vector inputs and a well-defined output

scale in order for the comparison to be legitimate.

TABLE I. ILLUSTRATIVE PERFORMANCE COMPARISON

Parameter	Conventional CORDIC	Proposed LUT-Assisted CORDIC
Input / output width	17-bit / 18-bit	17-bit / 18-bit
Fine CORDIC iterations	16	8
Coarse-angle LUT	Not used	16-entry example LUT
LUT utilization	520	455
Flip-flops	190	165
DSP blocks	0	0
Power consumption	0.078 W	0.066 W
Critical-path delay	22.8 ns	16.4 ns
Maximum operating frequency	43.86 MHz	60.98 MHz
Processing cycles (example)	16	9
Maximum angular error	≤ 0.20°	≤ 0.20°

The critical-path delay yields the maximum operating frequency. The chosen architecture as well as regulate sequence determine the number of iterations and processing cycles.

V. CONCLUSION AND FUTURE SCOPE

In this paper, an area-efficient CORDIC structure for fixed-point arc tangent computation with LUT assistance was described. The suggested datapath integrates angle combination, quadrant restoration, reduced-iteration CORDIC refinement, LUT-based coarse-angle estimation, input normalisation, and quadrant identification. In order to minimise the residual angle that the sequential shift-and-add unit processes, the architecture employs a rough starting estimate.With fixed-point angular accuracy, the suggested approach aims to minimise the amount of fine CORDIC iterations. A design trade-off among table size, remaining iteration count, equipment area, and numerical inaccuracy is introduced when a LUT is used. Verilog simulation, a Vivado synthesis, implementation, time analysis, and power calculation must all be used to determine the final performance.

REFERENCES

[1] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.
[2] J. S. Walther, “A unified algorithm for elementary functions,” in Proc. Spring Joint Computer Conference, 1971, pp. 379–385.
[3] Y. H. Hu, “CORDIC-based VLSI architectures for digital signal processing,” IEEE Signal Processing Magazine, vol. 9, no. 3, pp. 16–35, Jul. 1992.
[4] R. Andraka, “A survey of CORDIC algorithms for FPGA based computers,” in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays, 1998, pp. 191–200.
[5] B. Lakshmi and A. S. Dhar, “CORDIC architectures: A survey,” International Journal of Applied Mathematics and Computer Science, vol. 2010, Article ID 794891, 2010.
[6] P. K. Meher, J. Valls, T. B. Juang, K. Sridharan, and K. Maharatna, “50 years of CORDIC: Algorithms, architectures, and applications,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.
[7] H. Zhu, Y. Ge, and B. Jiang, “Modified CORDIC algorithm for computation of arctangent with variable iterations,” in Proc. IEEE 13th Int. Conf. Signal Processing (ICSP), 2016.
[8] V. Torres, J. Valls, and M. J. Canet, “Optimised CORDIC-based atan2 computation for FPGA implementations,” Electronics Letters, vol. 53, no. 19, pp. 1296–1298, Sep. 2017.
[9] C. J. Chih, K. L. Chen, and S. G. Chen, “A CORDIC processor with efficient table-lookup schemes for rotations and on-line scale factor compensations,” in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), 2005, pp. 3315–3318.
[10] “Real-time and high-accuracy arctangent computation using CORDIC and fast magnitude estimation,” Electronics, vol. 6, no. 1, Art. no. 22, 2017.
[11] J. Duprat and J. M. Muller, “The CORDIC algorithm: New results for fast VLSI implementation,” IEEE Transactions on Computers, vol. 42, no. 2, pp. 168–178, Feb. 1993.
[12] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2010.
[13] S. K. Mitra, Digital Signal Processing: A Computer-Based Approach, 4th ed. New York, NY, USA: McGraw-Hill, 2011.
[14] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. New York, NY, USA: Wiley, 1999.
[15] M. D. Ercegovac and T. Lang, Digital Arithmetic. San Francisco, CA, USA: Morgan Kaufmann, 2004.
[16] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed. New York, NY, USA: Oxford University Press, 2010.
[17] R. E. Blahut, Fast Algorithms for Signal Processing. Cambridge, U.K.: Cambridge University Press, 2010.
[18] R. G. Lyons, Understanding Digital Signal Processing, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2011.
[19] Xilinx, Vivado Design Suite User Guide: Synthesis (UG901). San Jose, CA, USA: Xilinx, current version applicable to the selected Vivado release.
[20] AMD Xilinx, Vivado Design Suite User Guide: Power Analysis and Optimization, UG907. San Jose, CA,

USA: AMD Xilinx, version applicable to the selected Vivado release.