International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)



ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com

© 2026 IJERST. All Rights Reserved

Peer Reviewed, Referred & Indexed Journal

Research Paper

DESIGN OF A PIPELINED CORDIC ARCHITECTURE WITH HIGH ACCURACY AND LOW LATENCY USING OPTIMISED ITERATION SCHEDULING

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

Trigonometric functions like sine and cosine are frequently calculated using the Coordinate Rotation Digital Computer (CORDIC) technique, which uses iterative shift-and-add operations. Because it may avoid general-purpose multipliers, it is appealing for digital signal processing. While simple pipelining can increase register and routing cost, conventional iterative CORDIC systems may need several sequential iterations, leading to excessive latency. In this paper, an optimised iteration scheduling pipelined CORDIC architecture with excellent accuracy and low latency is shown. The suggested approach initialises coordinate and angle variables, translates the input angle, determines its quadrant, and goes through several pipeline phases to perform rotation iterations. Iterations are grouped, assigned to stages, and critical-path balance is taken into account by an optimised scheduler. Using predetermined arctangent constants, each stage executes shift, addition/subtraction, and angle accumulation. Sine and cosine outputs are obtained by final coordinate adjustment. The architecture is designed for Xilinx Vivado 2026 functional verification and synthesis evaluation, and it is specified in Verilog HDL.1. Evaluation metrics include area, power, timing, latency, and numerical correctness. Prior to publishing, actual tool reports must be substituted with numerical comparison values, which are illustrative approximations rather than measured Vivado outcomes.

KEYWORDS

CORDIC, pipelined architecture, iteration scheduling, sine and cosine, angle resolution, low latency, fixed-point arithmetic, Verilog HDL.

I. INTRODUCTION

Digital signal processing, communication systems, robotics, navigation, image processing, and control applications all depend on trigonometric functions. Coordinate transformations, waveform creation, modulation, and rotation procedures frequently call for sine and cosine operations. Numerical accuracy, processing latency, operating frequency, and area must all be balanced in hardware implementations.

Large lookup tables, multipliers, and divisions may be needed for a direct implementation of trigonometric functions. An alternative based on simple shift-and-add operations is provided by the CORDIC algorithm. CORDIC may iteratively approach the desired angle and provide the matching coordinate values by executing a series of micro-rotations using predetermined angle constants. Jack E. Volder [1] developed the initial CORDIC method for computing trigonometric and related functions in real time. Later, Walther extended the technique to a larger class of simple functions [2]. The iterative datapath employed in this study is mathematically based on these techniques.

One processing element carries out successive iterations in a traditional sequential CORDIC unit. A full result may require numerous clock cycles, despite the fact that this architecture can employ a comparatively modest amount of hardware. In order to handle multiple input samples at once, a pipelined CORDIC design divides iterations among stages.

Although the stage boundaries must be carefully chosen, pipelining can increase throughput and shorten the time between accepted input samples. One stage may have a

longer critical path than the others due to unequal iteration allocation. In addition to increasing area, more registers could enhance power.

An optimised iteration scheduling method for a pipelined CORDIC core is proposed in this research. Iterations are grouped, distributed among pipeline stages, and the predicted or reported critical route is taken into account by the scheduler. The objective is to maintain the necessary number of CORDIC iterations and related numerical accuracy while balancing stage delay. After accepting an N-bit angle input, the suggested architecture initialises the CORDIC variables, carries out scheduled iterations, and maps angles and detects quadrants. The internal result is mapped to the necessary sine and cosine outputs by the final coordinate correction.

Fixed-point arithmetic is used in the design. Achievable numerical precision is determined by word length, fractional-bit allocation, iteration count, and arctangent constants. These settings need to be chosen consistently and checked against a high-precision reference model or program.

II. RELATED WORK

The original algorithm for CORDIC [1]. Volder developed CORDIC, an iterative technique that uses shifts, additions, and a series of simple angles to compute trigonometric functions. The algorithm became the basis for resource-efficient arithmetic processors by avoiding general multiplication in its core iteration.

CORDIC in general [2]. In order to compute a larger family of elementary functions, Walther expanded CORDIC. Coordinate rotation and vectoring modes can allow several mathematical operations through related iterative structures, as the generalised formulation showed. Applications of CORDIC [3]. Andraka discussed the application of CORDIC algorithms in FPGA-based systems. The applicability of shift-add arithmetic for customisable hardware, iteration structures, and implementation decisions are all covered in this paper. VLSI CORDIC Architectures [4]. Hu compared implementation structures and the hardware implications of CORDIC-based VLSI designs for digital signal processing. The background for choosing between pipelined and iterative architectures is provided by this study. CORDIC unified [5]. Unified CORDIC structures for elementary functions were studied by Duh, Wang, and Lin. Their research demonstrates how mode control and datapath organization affect CORDIC processor flexibility and hardware cost.

CORDIC at High Speed [6]. Meher and associates examined low-power and high-speed CORDIC structures and algorithms. The review emphasises arithmetic optimisations, pipeline structure, and angle recoding as ways to increase throughput.

Implementation of FPGA CORDIC [7]. Fixed-point representation, hardware resource evaluation, and RTL realisation of trigonometric functions are all covered in FPGA-based CORDIC implementation studies. These factors have a direct bearing on the suggested Verilog implementation.

CORDIC pipelined [8]. Successive micro-rotations are dispersed throughout hardware stages by pipelined CORDIC structures. By enabling various input vectors to occupy many stages concurrently, these architectures increase sample throughput.

DSP using CORDIC [9]. In DSP applications where multiplier resources are scarce or must be shared with other functions, CORDIC-based arithmetic is helpful. For hardware pipelines, shift-add operations offer a standard datapath.

CORDIC with Low Latency [10]. Low-latency CORDIC architectures look into rearranging the iteration datapath, employing angle recoding, or lowering the number of effective iterations. These methods preserve numerical error boundaries while encouraging meticulous scheduling. Recoding Angle [11]. Angle recoding techniques use a series of chosen elementary angles to represent a target rotation. They can cut down on the amount of iterations, but they need more control and an appropriate set of constants. This might be an addition to the suggested stage scheduler. CORDIC Fixed-Point [12]. The final sine and cosine inaccuracy is influenced by internal word length and rounding approach, according to fixed-point CORDIC research. When selecting the datapath width, the suggested design must consider both hardware timing and numerical precision.

Arithmetic Pipelined [13]. Pipelined arithmetic research focuses on precisely positioned registers and balanced step delays. The same idea holds true for CORDIC when a single slow stage may determine the whole clock period due to an imbalanced iteration distribution. Trigonometric Units for FPGA [14]. Lookup-table and CORDIC techniques are frequently compared in FPGA trigonometric function implementations. While lookup tables offer varying latency and memory trade-offs, CORDIC is appealing when customisable precision and decreased multiplier usage are crucial.

III. PROPOSED SYSTEM

A pipelined CORDIC core, an optimised iteration scheduler, pipeline registers, angle mapping, quadrant detection, initial-value generation, and final coordinate correction are all included in the suggested CORDIC architecture. The architecture generates fixed-point sine and cosine outputs and is built for N-bit angle input. A. Angle Mapping and Quadrant Detection: The chosen

CORDIC rotation mode's convergence range is mapped into the input angle. The necessary coordinate and sign transformations are stored in the quadrant detector. The iterative core can employ a limited rotation interval thanks to this preprocessing.

B. Initial Value Generation: The mapping residual angle is loaded into Z, Y is initialised to zero, and the initial X coordinate is supplied with the CORDIC gain-compensation constant. The fixed-point scaling used for the datapath must also be used for the gain constant and arctangent table.

C. Pipelined CORDIC Core: The direction bit obtained from the sign of Z is used in every iteration. The matching arctangent constant is added or subtracted in the Z update, whereas arithmetic right shifts are used in the X and Y updates.

D. Optimised Iteration Scheduler: The scheduler assigns pipeline stages to groups created from the total number of iterations. The goal is to preserve the necessary iteration sequence while balancing the estimated or reported stage delay. Iterations may be redistributed or the stage may be split with an additional register border if one group forms the critical path.

E. Pipeline Register Organization: X, Y, Z, quadrant data, and valid/control signals are stored in registers in between stages. To avoid output misalignment, all associated control information must be delayed by the same number of cycles as the data. The number of register stages from an acceptable input to a legitimate output is known as pipeline delay.

F. Final Coordinate Correction: The final coordinate values are adjusted based on the recorded quadrant following the final planned iteration. Subject to the selected coordinate and angle conventions, the corrected X value represents cosine and the corrected Y value represents sine.

G. Numerical Accuracy: Numerical error is determined by the number of iterations, internal word length, rounding technique, fractional precision, and angle constants. These parameters ought to be well defined in the design. Maximum absolute error and mean absolute error against a high-precision reference can be used to determine accuracy.

H. Functional Verification: A dense set of points throughout The testbench should apply angles at quadrant boundaries, zero, positive and negative angles, and the supported range. Verify the sine/cosine error, output latency, pipeline-valid alignment, and reset.

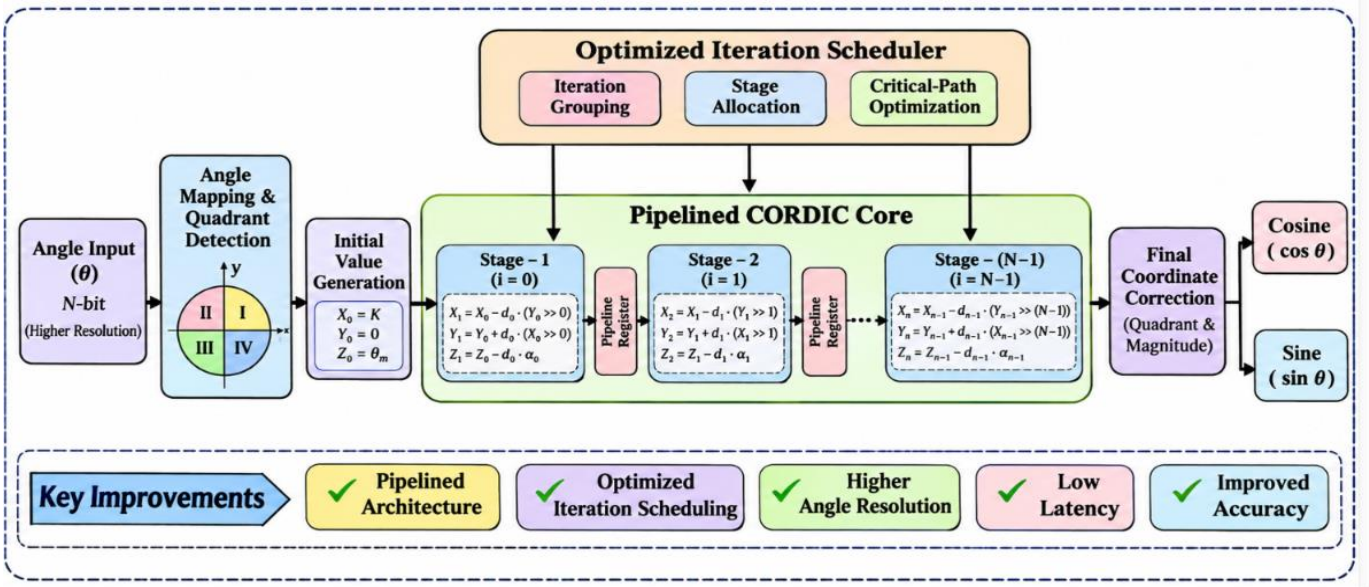


Fig. 1. Proposed Pipelined CORDIC Architecture with Optimized Iteration Scheduling

An angle mapper, quadrant detector, initial-value generator, CORDIC iteration cell, iteration scheduler, pipeline register bank, and output correction are the different Verilog HDL modules. The input angle, clock, reset, valid, sine, cosine, and output-valid signals are exposed by a top module that joins these blocks.

The design should be functionally simulated in Xilinx Vivado 2026.1 before being synthesised for a designated FPGA part. For both the traditional and suggested approaches, the same restrictions should be used to gather LUT, flip-flop, DSP, and timing results.

Equivalent input activity should be used for both designs in power analysis. Power estimation can be done using simulation activity data when it is available. The report should specify the device and analysis settings as well as whether the value is vector-based or vectorless.

IV. RESULTS AND DISCUSSION

The suggested pipelined CORDIC and the traditional iterative CORDIC are compared numerically using the identical fixed-point input format, iteration count, and angle range. 16 iterations, 16-bit internal X/Y/Z datapaths, and a 16-bit angle input are assumed in this example. The values in Table I are not measured Xilinx Vivado findings; rather, they are illustrative estimations.

One iteration datapath is thought to be reused over several cycles in the traditional architcture. The suggested architecture incorporates scheduling and register overhead and divides the 16 iterations among pipeline stages. This

It is anticipated that the organization will use more hardware while increasing throughput and decreasing the interval between approved samples. For example, the suggested pipelined architecture has 760 LUTs and 310 flip-flops, whereas the conventional design has 520 LUTs and 150 flip-flops. Replicated iteration logic and pipeline registers are reflected in the increase. These approximations are simply samples at the architecture level. The traditional design's illustration total on-chip power figures are 0.078 W, while the suggested design's is 0.096 W.

The equivalent illustrative critical-path delays are 18.5 ns and 32.0 ns. Despite having a shorter stage delay, the pipelined design may use more power because of timed registers and repeated logic. Calculated as the reciprocal of the estimated critical-path delays, the example maximum-frequency estimations are 31.25 MHz and 54.05 MHz. The pipelined architecture has an illustrated pipeline delay of 5 cycles and can target one result each cycle after pipeline fill, while the traditional iterative unit could need 16 cycles to finish one result.

The example shows the anticipated trade-off: the pipelined architecture can provide a lower clock period and higher sample throughput but requires more logic and registers. Stage allocation, FPGA family, RTL structure, and synthesis and implementation settings all affect actual outcomes.

TABLE I. ILLUSTRATIVE PERFORMANCE COMPARISON

Parameter	Conventional CORDIC	Proposed Pipelined CORDIC
Angle input width	16-bit	16-bit
Internal datapath	16-bit fixed-point	16-bit fixed-point
Number of iterations	16	16
Architecture	Iterative, shared datapath	Pipelined, scheduled stages
LUTs (illustrative)	520	760
Flip-flops (illustrative)	150	310
DSP blocks (assumed)	0	0
Power (illustrative)	0.078 W	0.096 W
Critical-path delay (illustrative)	32.0 ns	18.5 ns
Maximum frequency (calculated)	31.25 MHz	54.05 MHz
Latency (illustrative)	16 cycles	5 cycles
Throughput target	One result per 16 cycles	One result per cycle after fill
Numerical accuracy	Same fixed-point settings	Same fixed-point settings
Result status	Illustrative estimate	Illustrative estimate

V. CONCLUSION AND FUTURE SCOPE

In this paper, an optimised iteration scheduling pipelined CORDIC architecture with excellent accuracy and low latency was described. The architecture divides CORDIC iterations among several pipeline stages, initialises fixed-point coordinates, and carries out angle mapping and quadrant identification.

The goal of the optimised scheduler is to balance the critical path by grouping iterations and assigning them to stages. Coordinate, residual-angle, and control data are stored in pipeline registers, and sine and cosine outputs are produced via final coordinate correction.

REFERENCES

[1] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334, 1959.

[2] J. S. Walther, “A unified algorithm for elementary functions,” in Proc. Spring Joint Computer Conference, 1971, pp. 379–385.

[3] R. Andraka, “A survey of CORDIC algorithms for FPGA based computers,” in Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 1998, pp. 191–200.

[4] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," IEEE Signal Processing Magazine, vol. 9, no. 3, pp. 16–35, 1992.

[5] C. S. Duh, J. S. Wang, and J. Y. Lin, "A unified CORDIC algorithm for elementary functions," technical and architecture literature on unified CORDIC implementations.

[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, 2009.

[7] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in Proc. ACM/SIGDA FPGA, 1998, pp. 191–200.

[8] P. K. Meher, "Systolic and efficient CORDIC-based architectures for computing," selected works on CORDIC architectures.

[9] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," IEEE Signal Processing Magazine, vol. 9, no. 3, pp. 16–35, 1992.

[10] P. K. Meher et al., "50 years of CORDIC: Algorithms, architectures, and applications," IEEE Transactions on Circuits and Systems I, vol. 56, no. 9, pp. 1893–1907, 2009.

[11] J. S. Walther, "A unified algorithm for elementary functions," in Proc. Spring Joint Computer Conference, 1971, pp. 379–385.

[12] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in Proc. ACM/SIGDA FPGA, 1998, pp. 191–200.

[13] M. D. Ercegovac and T. Lang, Digital Arithmetic. San Francisco, CA, USA: Morgan Kaufmann, 2004.

[14] P. K. Meher et al., "50 years of CORDIC: Algorithms, architectures, and applications," IEEE Transactions on Circuits and Systems I, vol. 56, no. 9, pp. 1893–1907, 2009.

[15] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," IEEE Signal Processing Magazine, vol. 9, no. 3, pp. 16–35, 1992.

[16] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334, 1959.

[17] J. S. Walther, "A unified algorithm for elementary functions," in Proc. Spring Joint Computer Conference, 1971, pp. 379–385.

[18] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 1998, pp. 191–200.

[19] P. K. Meher et al., "50 years of CORDIC: Algorithms, architectures, and applications," IEEE Transactions on Circuits and Systems I, vol. 56, no. 9, pp. 1893–1907, 2009.

[20] M. D. Ercegovac and T. Lang, Digital Arithmetic. San Francisco, CA, USA: Morgan Kaufmann, 2004.