Design and implementation of an Adaptive Pipelined CORDIC Architecture for Wireless Communication System on FPGA Platform

Dr. Shashi Raj K[1], Dr. Ajay Kumar[2]

[1]Postdoctoral Researcher, Lincoln University College, Malaysia. Email ID: [1]pdf.shashiraj@lincoln.edu.my

[2]Lincoln University College, Malaysia. Email ID: pdfsv.ajaykumar@lincoln.edu.my

[2]IILM University, Greater Noida, India. Email ID: ajay.phdcse@gmail.com

---

Abstract: An adaptive pipelined Coordinate Rotation Digital Computer (CORDIC) architecture for high-speed FPGA-based Wireless communication applications is presented in this study. A 16-bit fixed-point datapath and an adaptive iteration control mechanism are used in the suggested architecture, which dynamically modifies the number of active CORDIC iterations in accordance with the necessary computing accuracy. The suggested design uses adaptive iteration and bypass logic to remove unnecessary calculations while keeping a fixed pipeline depth, in contrast to traditional pipelined CORDIC implementations. The design is implemented using Xilinx ISE 14.7 on an Artix-7 FPGA and written in Verilog HDL. A maximum operating frequency of 420.026 MHz, a throughput of 480.029 Mbps, and hardware usage of 672 slices and 1709 LUTs are all shown by experimental results. Functional verification confirms that the suggested design offers an effective trade-off among speed, area, power, and numerical accuracy for real-time DSP and wireless communication systems, achieving around 99.99% computational accuracy.

Keywords: Adaptive CORDIC, FPGA, Pipelined Architecture, Wireless Communication.

Introduction

An effective iterative method for integrating basic mathematical operations into digital circuitry is the Coordinate Rotation Digital Computer (CORDIC) algorithm. It eliminates the need for hardware multipliers by performing vector rotation through a series of predetermined micro-rotations using just arithmetic addition, subtraction, and binary shift functions. For computing trigonometric, hyperbolic, logarithmic, exponential, multiplication, division, and inverse trigonometric operations in digital signal processing (DSP), wireless networks, and embedded systems, CORDIC is frequently used in FPGA- and ASIC-based implementations because of its straightforward architecture [1], [2].

The traditional CORDIC method has comparatively significant latency despite its hardware cost, since its computational correctness is dependent on the number of repetitions. An N-bit correct output usually necessitates N iterations, which limits throughput and lengthens execution time. Computational complexity is further increased by the need for an extra compensation stage due to the accumulated scaling factor throughout subsequent rotations. Several improved CORDIC designs, such as pipelined, parallel, scale-free, and adaptive implementations, have been documented in the

literature [3], [4] to get around these restrictions. Among these methods, pipelined CORDIC systems greatly increase throughput by processing a large number of samples at once. Conventional pipelined architectures, on the other hand, run all iterations regardless of the requisite accuracy, which results in more calculations and higher dynamic power usage. By activating just, the necessary number of CORDIC stages and bypassing the other stages, adaptive iteration approaches get around this issue and increase hardware efficiency without changing pipeline depth.

Related works

This section discusses a few of the current CORDIC work. A CORDIC-based architecture for calculating arcsine and arccosine functions with enhanced gain compensation was presented by Paz and Garrido et al. [5]. The suggested solution delivers improved accuracy without adding stage complexity and has been tested effectively on an FPGA, in contrast to traditional approaches that need double iterations or display substantial approximation errors. A simplified radix-4 hyperbolic CORDIC architecture for calculating the N^{th} root and N^{th} power of fixed-point numbers was introduced by Changela et al. [6]. The solution, which was implemented on a Virtex-6 FPGA, improved numerical accuracy over traditional radix-2 CORDIC implementations while reducing hardware use by 19%.

An FPGA-based CORDIC design was constructed by Zhang et al. [7] to generate arbitrary phase-varying digital chirp signals (DCS). The suggested architecture effectively produces both linear and nonlinear chirp signals with increased precision by combining an expanded CORDIC core with a trajectory approximator to reduce phase errors. The Uniformly Distributed (UD) CORDIC method was first presented in binary and Canonical Signed Digit (CSD) formats by Garrido et al. [8]. When compared to traditional CORDIC designs, FPGA and ASIC execution showed gains in area efficiency, operation frequency, and power usage. An approximate-adder-based CORDIC framework was presented by Luyen et al. [9] to implement the Izhikevich neuron model. The design maintained a low mean relative error of 0.07% while increasing throughput by 14.56% and reducing area by 10.23% by substituting approximate arithmetic units for accurate adders.

Method

The suggested adaptive pipelined CORDIC architecture uses a fixed-point shift-and-add procedure to efficiently rotate vectors. The suggested design adds an adaptive iteration strategy via the iter_ctrl input, in contrast to traditional CORDIC versions that always carry out the maximum number of iterations. The number of active CORDIC iterations is constantly determined by this control signal based on the necessary computational precision. As an outcome, the rotation operation is only carried out by the appropriate pipeline stages, with the remaining stages operating in bypass mode. This reduces needless switching activity and increases hardware efficiency without changing the pipeline stage as a whole. An input register stage, twelve identical pipelined CORDIC stages, an adaptive iteration controller, an arctangent lookup table (LUT), and an output register stage make up the architecture seen in Figure 1. The input rotation angle (θ_{in}) and input vector (x_{in}, y_{in}) are sampled and placed in the first pipeline registers during each clock cycle. These registered values provide synchronous data transmission across the pipeline and set up the CORDIC calculation.

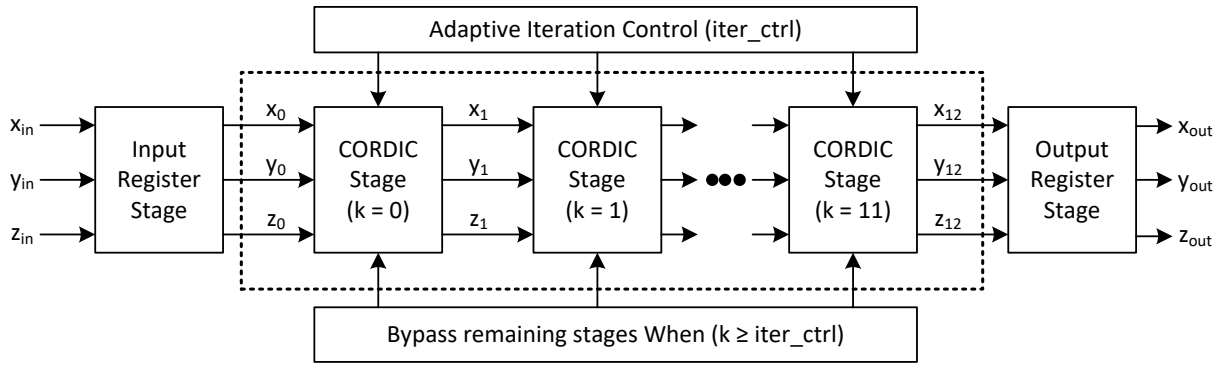


Figure 1. Proposed adaptive pipelined CORDIC architecture

The initialization can be stated mathematically as $x_0 = x_{in}, y_0 = y_{in},\ z_0 = \theta_{in}$. Each pipeline step uses arithmetic right-shift procedures to accomplish a single elementary CORDIC rotation instead of using hardware multipliers. Equation (1) is used to get the shifted values for the k^{th} iteration. where multiplication by 2^{-k} is implemented by the shift operator. The rotational direction is determined by the sign of the residual angle z_k. Equation (2) states that the vector is rotated clockwise if $z_k \geq 0$. In contrast, the vector is rotated in the opposite direction as shown in Equation (3) when $z_k < 0$. These iterative equations provide the necessary rotated vector while progressively reducing the remaining angle.

$$x_{shift} = x_k \gg k,\ y_{shift} = y_k \gg k, \tag{1}$$

$$x_{k+1} = x_k - y_{shift} y_{k+1} = y_k + x_{shift};\ z_{k+1} = z_k - \tan^{-1}\left(2^{-k}\right) \tag{2}$$

$$x_{k+1} = x_k + y_{shift}; y_{k+1} = y_k - x_{shift};\ z_{k+1} = z_k + \tan^{-1}\left(2^{-k}\right). \tag{3}$$

The suggested design keeps all primary rotation angles in a Q1.15 fixed-point arctangent lookup table (LUT) in order to remove the computational complexity related to trigonometric operations. The residual angle is updated at each iteration by directly accessing the relevant value of $\tan^{-1}\left(2^{-k}\right)$ from the LUT. The adaptive iteration controller, which matches the current pipeline stage index k with the user-defined iteration count iter_ctrl, is an essential part of the suggested architecture. The stage does the standard CORDIC rotation when the condition $k <$ iter_ctrl is met. If not, Equation (4) merely propagates the current values to the subsequent step, disabling the arithmetic unit.

$$x_{k+1} = x_k,\ y_{k+1} = y_k,\ z_{k+1} = z_k \tag{4}$$

By avoiding unnecessary computations during the idle phases, this bypass technique lowers dynamic power usage while preserving a steady pipeline latency. Timing management doesn't require any additional control circuitry because all pipeline stages stay in sync with the system clock. Dedicated pipeline registers that hold the updated values of x, y, and z come after each CORDIC stage. These registers enable all pipeline stages to work simultaneously on various input vectors by isolating the combinational delay of neighboring stages. As a result, once the pipeline is fully filled, each clock cycle produces a new output vector, resulting in high throughput appropriate for wireless communication usage. Lastly, the output register stage collects the outputs from the last pipeline step to provide the

residual angle z_{out} and the final rotated coordinates (x_{out}, y_{out}). The deterministic time behavior results from a fixed pipeline depth even while the number of active iterations vary based on iter_ctrl.

Results and Discussion

The suggested adaptive pipelined CORDIC architecture was constructed on a Artix-7 FPGA using Xilinx ISE 14.7 and functionally validated using Modelsim simulation. Functional correctness, computational accuracy, FPGA resource usage, operating frequency, power usage, latency, and throughput were the metrics used to assess the architecture's performance.

The functional simulation waveform of the suggested adaptive pipelined CORDIC architecture, which is implemented in Verilog HDL, is shown in Figure 2. The reset signal sets all pipeline registers to zero during the first clock cycles (CCs). The input vector is loaded with x_{in}=19898, y_{in}= 0, and an input angle of 45°, which is represented as 25735 in Q1.15 fixed-point format, following the deassertion of reset. The architecture is set up to use 12 CORDIC iterations (iter_ctrl = 12) to carry out the maximum precision mode. The calculated outputs converge to x_{out} =23167, y_{out}= 23174, and a residual angle $z_{out} = -3$ after the pipeline latency of 14 CCs, indicating successful vector rotation. The adaptive functionality of the suggested architecture is then demonstrated by changing the input angle to 12867 and reducing the iteration count to 8 (iter_ctrl = 8). The waveform confirms that the architecture appropriately supports adaptive iteration control while keeping the pipeline depth constant and generating a single output each time the pipeline is full.

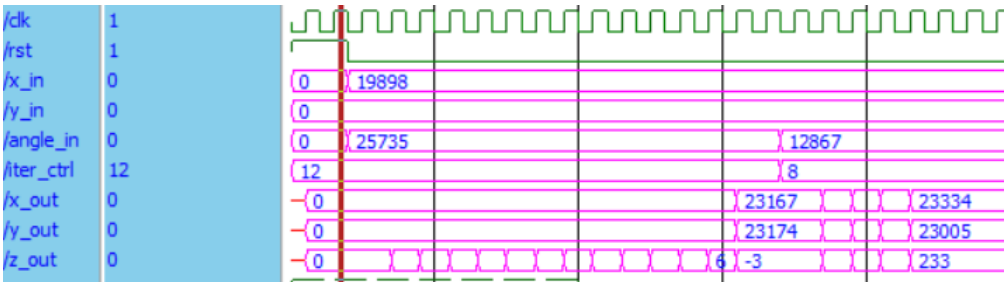


Figure 2. Simulation results of Adaptive Pipeline CORDIC architecture

Table 1 shows the numerical accuracy of the suggested adaptive pipelined CORDIC design by comparing the simulated outputs with the anticipated theoretical values for both positive and negative 45° rotations. The Q1.15 implementation's fixed-point quantization and finite word-length format are mainly responsible for the maximum absolute inaccuracy of just 5 LSBs. As a result, for every test scenario, the percentage error stays below 0.02%, and the computed accuracy is about 99.98–99.99%. These findings show that the suggested adaptive architecture is appropriate for high-accuracy communication usage as it maintains exceptional numerical precision even when using simply shift-and-add operations and adaptive iteration control.

Table 1. Estimation of % error and accuracy results of Adaptive Pipeline CORDIC architecture

Angle	Expected Output	Simulated Output	Absolute Error	% Error	% Accuracy
$+45^{\circ}$ (X)	23170	23167	3	0.01	99.99
$+45^{\circ}$ (Y)	23170	23174	4	0.02	99.98
-45° (X)	23170	23174	4	0.02	99.98

-45° (Y)	-23170	-23165	5	0.02	99.98

The FPGA implementation outcomes of the suggested adaptive pipelined CORDIC architecture generated on the Xilinx Artix-7 FPGA are summarized in Table 2. The architecture uses a 16-bit fixed-point datapath with 12 CORDIC rounds, with 672 slices, 1709 LUTs, and 576 flip-flops and suggesting moderate hardware complexity.

Table 2. Resource usage of proposed architecture on Artix-7

Resources	Usage	Resources	Usage
Data width	16	Dynamic Power (mW)	86
Iterations	12	Total Power (mW)	168
Slices	672	Latency (CC)	14
LUTs	1709	Throughput (Mbps)	480.029
LUT-FFs	576	Efficiency (Mbps/Slice)	0.71432
Max. Frequency (MHz)	420.026	Efficiency (Mbps/LUT)	0.2808
Static Power (mW)	82	% Accuracy	99.99

The design uses 82 mW of static power and 86 mW of dynamic power to reach a maximum operating frequency of 420.026 MHz, with a total power dissipation of 168 mW. Due to continuous pipeline operation, the fully pipelined system maintains a high throughput of 480.029 Mbps while introducing a Latency of 14 CCs. Additionally, the hardware efficiency is assessed at 0.714 Mbps/Slice and 0.281 Mbps/LUT, indicating a successful balance between FPGA resource usage and computational performance.

Conclusion

An adaptive pipelined CORDIC design with adaptive iteration control has been effectively designed and implemented for FPGA-based wireless applications. The suggested architecture reduces unnecessary calculations while maintaining a consistent pipeline latency and adequate throughput by just activating the relevant CORDIC stages and bypassing the remaining pipeline stages. A maximum operating frequency of 420.026 MHz, throughput of 480.029 Mbps, and exceptional computational accuracy of almost 99.99% with minimum numerical error are all obtained by the implementation on the Artix-7 FPGA. Additionally, the architecture shows effective use of FPGA resources by using just 672 slices and 1709 LUTs. The obtained outcomes confirm that the suggested adaptive pipelined CORDIC architecture provides an efficient trade-off between computational accuracy, power usage, operating speed, and hardware complexity, making it appropriate for contemporary FPGA-based DSP, wireless networks, and embedded computing tasks.

References

[1] L. Vachhani, K. Sridharan, and P. K. Meher, “Efficient CORDIC algorithms and architectures for low area and high throughput implementation,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 1, pp. 61–65, Jan. 2009.

[2] R. Andraka, “A survey of CORDIC algorithms for FPGA based computers,” in *Proc. ACM/SIGDA Int.*

Symp. FPGAs, Feb. 1998, pp. 191–200.

[3] Y. H. Hu, ‘‘CORDIC-based VLSI architectures for digital signal processing,’’ *IEEE Signal Process. Mag.*, vol. 9, no. 3, pp. 16–35, Jul. 1992.

[4] H.-T. Nguyen, X.-T. Nguyen, T.-T. Hoang, D.-H. Le, and C.-K. Pham, ‘‘Low-resource low-latency hybrid adaptive CORDIC with floating-point precision,’’ *IEICE Electron. Exp.*, vol. 12, no. 9, 2015, Art. no. 20150258.

[5] P. Paz and M. Garrido, "CORDIC-Based Computation of Arcsine and Arccosine Functions on FPGA," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 9, pp. 3684-3688, Sept. 2023, doi: 10.1109/TCSII.2023.3262353.

[6] A. Changela, Y. Kumar, M. Woźniak, J. Shafi, and M. F. Ijaz, “Radix-4 CORDIC algorithm based low-latency and hardware efficient VLSI architecture for Nth root and Nth power computations,” *Sci. Rep.*, vol. 13, no. 1, p. 20918, 2023, doi: 10.1038/s41598-023-47890-3.

[7] H. Zhang, M. Chen, S. Wan, Z. Jiao, S. Hu and P. Song, "An Arbitrary-Phase-Varying Digital Chirp Signal Generator Based on Improved CORDIC Algorithm and Trajectory Approximator," in *IEEE Access*, vol. 12, pp. 135362-135371, 2024, doi: 10.1109/ACCESS.2024.3456554.

[8] M. Garrido, D. Medina, P. Paz and M. López-Vallejo, "Uniformly Distributed CORDIC," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 11, pp. 6948-6961, Nov. 2025, doi: 10.1109/TCSI.2025.3560802.

[9] V. V. Luyen *et al*., "Energy-Efficient Izhikevich Neuron Design Using Approximate CORDIC-Based Multipliers for Low-Power Neuromorphic Hardware," in *IEEE Access*, vol. 14, pp. 22146-22161, 2026, doi: 10.1109/ACCESS.2026.3662681.