

# Design and Verification of a Pipelined CORDIC Processor for Sine and Cosine Generation

Mohan Krishna Sikhakolli

Department of Electronics and Communication Engineering  
Rajiv Gandhi University of Knowledge Technologies, Nuzvid  
mohankrishnasikhakolli@gmail.com

20 September 2025

## Abstract

This paper details the design, implementation, and verification of a 9-bit pipelined CORDIC (Coordinate Rotation Digital Computer) processor for generating sine and cosine values using eSim, an open-source EDA tool. The CORDIC algorithm is chosen for its hardware efficiency, as it computes trigonometric functions using only shifts and additions, making it ideal for FPGA and ASIC implementations. The architecture is fully pipelined with a depth of nine stages, enabling high-throughput calculations suitable for real-time digital signal processing (DSP) applications. This 9-bit implementation was specifically chosen as a trade-off between computational speed and resource utilization, dictated by project time constraints and the need for integration within a larger mixed-signal system. While this results in lower precision, it serves as a verified foundation for a future migration to a more accurate 18-bit design. Simulation results from iVerilog and GTKWave, managed within the eSim environment, are presented to validate the functionality of the core.

---

## 1 Introduction

The efficient calculation of trigonometric functions is a fundamental requirement in many fields, including telecommunications, graphics rendering, control systems, and in emerging AI applications for implementing activation functions. While software libraries can provide these functions, hardware-based solutions are often necessary for applications demanding high speed and parallelism. The CORDIC algorithm, first described by Jack Volder, offers an iterative, multiplier-less method to perform vector rotations, making it exceptionally well-suited for hardware implementation.

This project, developed entirely using the open-source eSim platform, focuses on the "Rotation Mode" of the CORDIC algorithm to compute the sine and cosine of a given input angle. A pipelined architecture is employed to maximize throughput. After an initial latency of nine clock cycles, the core can produce a new sine and cosine pair on every subsequent clock cycle. The design uses a 9-bit fixed-point representation to balance performance with the logic footprint, a crucial consideration for mixed-signal designs where digital resources can be limited.

## 2 CORDIC Algorithm and Fixed-Point Representation

### 2.1 Theoretical Principle

The CORDIC algorithm in rotation mode rotates a starting vector  $(x_0, y_0)$  by a target angle,  $\theta$ , through a series of smaller, iterative rotations. The key insight is that these micro-rotations are chosen to have angles whose tangents are powers of two ( $\arctan(2^{-i})$ ), allowing the rotational multiplications to be implemented as simple bit-shifts.

The iterative equations are:

$$\begin{aligned} x_{i+1} &= x_i - d_i \cdot y_i \cdot 2^{-i} \\ y_{i+1} &= y_i + d_i \cdot x_i \cdot 2^{-i} \\ z_{i+1} &= z_i - d_i \cdot \alpha_i \end{aligned}$$

where:

- $i$  is the iteration number, from 0 to N-1.
- $z_i$  is the residual angle to be rotated.
- $\alpha_i = \arctan(2^{-i})$  are the pre-computed elementary angles.
- $d_i$  is the direction of rotation: +1 if  $z_i$  is negative, and -1 if  $z_i$  is positive.

To compute sine and cosine, we initialize the vector with  $y_0 = 0$  and  $x_0$  set to a specific constant. After N iterations, the final vector will be:

$$\begin{aligned} x_N &\approx K(\cos(\theta)) \\ y_N &\approx K(\sin(\theta)) \end{aligned}$$

The term **K** is the gain inherent to the CORDIC algorithm, which arises because each micro-rotation slightly increases the vector's magnitude. It is a constant value that depends only on the number of iterations (N).

$$K = \prod_{i=0}^{N-1} \sqrt{1 + 2^{-2i}}$$

For a large number of iterations,  $K \approx 1.647$ .

### 2.2 Fixed-Point Calculations

To use the CORDIC algorithm for direct sine and cosine calculation, the starting vector  $(x_0, y_0)$  is initialized to  $(1/K, 0)$ . This pre-scaling cancels out the algorithm's gain, yielding final outputs of  $(x_N, y_N) = (\cos(\theta), \sin(\theta))$ .

- **Data Representation:** All inputs and outputs are 9-bit signed fixed-point numbers. The range of a 9-bit signed integer is  $[-256, 255]$ .
  - **Angle:** The input angle is mapped from degrees to this integer range. The range  $[-180^\circ, +180^\circ]$  is mapped to  $[-256, 255]$ . The conversion is:
 
$$\text{Fixed Point Angle} = \text{round}((\text{Angle in Degrees} / 180) * 256)$$
  - **Sine/Cosine:** The output values are mapped from the range  $[-1.0, +1.0]$  to  $[-256, 255]$ . For example, a value of 1.0 is represented as 255.
- **Initial X Value (1/K):** For our 9-bit system, the starting value  $x_{start}$  must be the fixed-point representation of  $1/K \approx 1/1.647 \approx 0.60725$ .

$$x\_start = \text{round}(0.60725 * 2^8) = \text{round}(155.456) = 155$$

This value, 9'd155, is assigned to `tb_x_start` in the testbench.

### 3 Hardware Implementation

The CORDIC processor was implemented in Verilog within the eSim design environment. It consists of a 9-stage pipeline, where each stage performs one iteration of the algorithm.

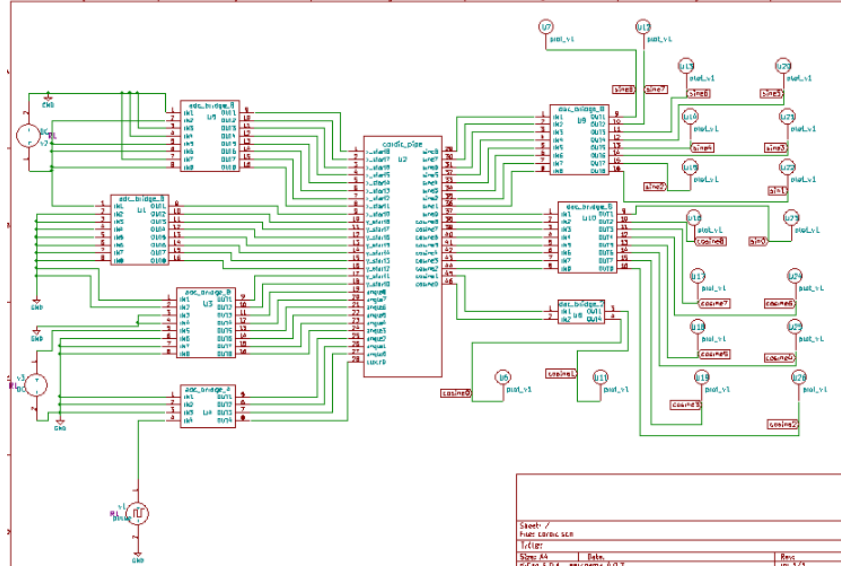


Figure 1: Schematic diagram of the CORDIC implementation.

- **atan\_table:** A constant look-up table stores the pre-calculated fixed-point values of the elementary angles,  $\arctan(2^{-i})$ .
- **Pipeline Registers:** Three sets of registers,  $x[0:8]$ ,  $y[0:8]$ , and  $z[0:8]$ , are used to pass the intermediate results from one stage to the next on each rising edge of the clock.
- **CORDIC Stage:** A **generate** loop creates the nine stages. Inside each stage, combinational logic performs the required operations:
  1. The direction of rotation is determined by the most significant bit (sign bit) of the incoming residual angle,  $z[i][8]$ .
  2. The  $x$  and  $y$  values are right-shifted ( $\gg$ ) by the stage index  $i$  to perform the division by  $2^i$ .
  3. Adders and subtractors update the  $x$ ,  $y$ , and  $z$  values based on the rotation direction.
- **Inputs/Outputs:** The module takes an initial vector ( $x\_start$ ,  $y\_start$ ), a target angle ( $angle$ ), and a clock signal. It outputs the final sine and cosine values from the last pipeline stage.

### 4 Simulation Results and Discussion

The design was simulated using the open-source eSim platform, which integrates iVerilog for simulation and GTKWave for waveform analysis. The clock period was set to 10 ns. The testbench applies a sequence of input angles and waits for the results to propagate through the 9-stage pipeline.

Time(ns)	Angle_in	Cosine_out	Sine_out
110	0	255	1
200	128	1	255
290	64	178	182
380	-128	1	-256
470	43	222	129

Figure 2: Console output from the iVerilog simulation

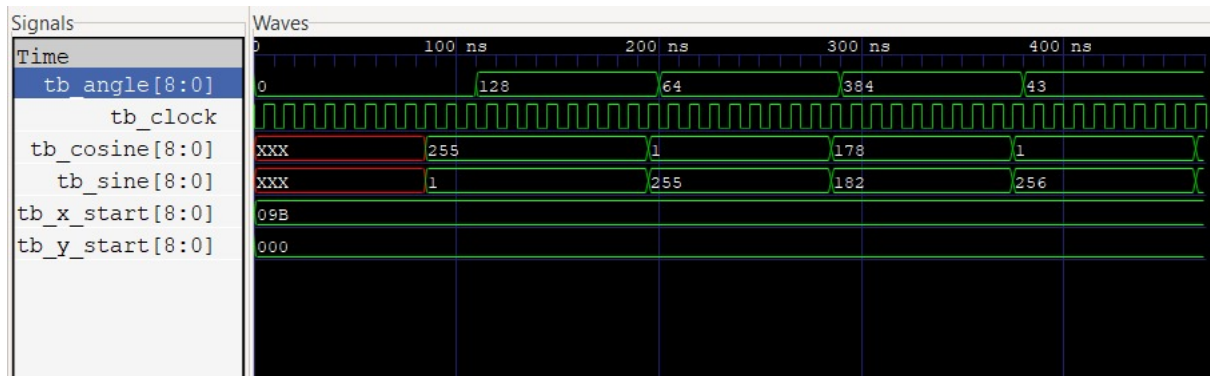


Figure 3: Digital simulation waveforms viewed in GTKWave, showing inputs and outputs over time.

Table 1: Comparison of Expected vs. Simulated CORDIC Outputs

Input Angle (Degrees)	Fixed-Point Angle Input	Expected Cosine	Simulated tb_cosine	Expected Sine	Simulated tb_sine
0°	0	255	255	0	1
90°	128	0	1	255	255
45°	64	181	178	181	182
-90°	-128	0	1	-256	-256
30.23°	43	221	222	129	129

#### 4.1 Analysis of Results

The following table compares the expected values with the simulated hardware outputs.

The results closely match the theoretical values. The minor discrepancies are due to the quantization errors inherent in a 9-bit fixed-point system and the finite number of iterations.

#### 4.2 Accuracy Limitations

The 9-bit datapath was a deliberate design choice to create a lightweight and fast core suitable for integration with other analog and digital components under tight project deadlines. However, this choice introduces accuracy limitations from two main sources:

1. **Quantization Error:** Representing angles and trigonometric values with only 9 bits limits their precision.

```

ngspice -p C:\Users\saiva\Sim-Workspace\cordic\cordic.cir.out
sine=182
cosine=178
=====cordic_pipe : New Iteration=====
Instance : 0

Inside foo before eval.....
x_start=155
y_start=0
angle=64
clock=1
sine=182
cosine=178

Inside foo after eval.....
x_start=155
y_start=0
angle=64
clock=1
sine=182
cosine=178
=====cordic_pipe : New Iteration=====
Instance : 0

```

Figure 4: Mixed-signal simulation output showing digital inputs and resulting analog sine/cosine waveforms.

2. **Truncation Error:** Halting the CORDIC algorithm after only 9 iterations means the final residual angle  $z[8]$  is not exactly zero, introducing a small error in the final rotation.

## 5 Future Work: Migration to 18-Bit

The logical next step is to migrate this verified 9-bit design to a more precise 18-bit implementation. An 18-bit datapath would offer a significant improvement in accuracy, making the core suitable for a much wider range of DSP applications.

This migration would involve:

- **Widening Datapaths:** All registers, wires, and arithmetic units would be extended to 18 bits.
- **Expanding the atan\_table:** The look-up table values would be re-calculated for 18-bit precision.
- **Increasing Iterations:** To take full advantage of the increased precision, the number of pipeline stages (iterations) would likely be increased to 18.

This successfully tested 9-bit module provides a scalable and robust foundation for the future development of the higher-precision 18-bit core.

## 6 Conclusion

This paper has presented the design and verification of a 9-bit, 9-stage pipelined CORDIC processor using the eSim open-source EDA tool. The Verilog implementation was shown to be functional, correctly calculating sine and cosine values within the expected precision limits of its fixed-point architecture. The choice of a 9-bit design was a pragmatic trade-off, prioritizing

implementation speed and resource constraints over numerical accuracy. The results confirm that the core is a viable solution for applications where high throughput is critical and moderate precision is acceptable, and demonstrates the power of open-source tools for complex digital design. Furthermore, this project serves as an essential proof-of-concept for a planned migration to a high-precision 18-bit architecture.