Project Report: 16-bit CORDIC Circuit Design and Simulation
Author: Mohd Maaz Quraishi
Date: October 27, 2025
Competition: eSim Hackathon 2025

---

1. Introduction

This report details the design and simulation of a 16-bit CORDIC (Coordinate Rotation Digital Computer) circuit implemented in Verilog.
The "16-bit" designation refers to the primary data path width used for coordinate calculations, defined by the `WIDTH` parameter.
The goal was to create a hardware-efficient module for calculating sine and cosine using only adders, subtractors, and bit-shifters, avoiding hardware multipliers.

Functional verification was performed using the Makerchip IDE.
Physical implementation targeting the IHP SG13G2 PDK was not completed due to time constraints.

---

2. CORDIC Algorithm Overview

The CORDIC algorithm iteratively rotates a 2D vector towards a target angle using pre-determined micro-rotations.
This implementation uses **Rotation Mode**, where an initial vector (x_0, y_0) is rotated by angle z_0.

The core equations are:
$x_{(i+1)} = x_i - d_i \cdot y_i \cdot 2^{-i}$
$y_{(i+1)} = y_i + d_i \cdot x_i \cdot 2^{-i}$
$z_{(i+1)} = z_i - d_i \cdot \alpha_i$

Where:
- i = iteration index (0 to 15)
- $\alpha_i = \arctan(2^{-i})$ are LUT constants
- $d_i = \mathrm{sign}(z_i)$ is the rotation direction

After N iterations:
$(x_N, y_N) \approx K \cdot (x_0 \cos z_0 - y_0 \sin z_0), K \cdot (y_0 \cos z_0 + x_0 \sin z_0)$

where the gain **$K \approx 1.647$**.

For cosine and sine calculation, we start with:
$x_0 = 1 / K$
$y_0 = 0$

---

3. Verilog Implementation

The circuit was implemented as two modules:

- **cordic.v**
- **angle_lut.v**

Both were instantiated and connected in Makerchip (`cordic_makechip_code.v`).

3.1 Key Features

- **Data Representation & Scaling (16-bit):** Defined by `WIDTH = 16`, using signed `[WIDTH-1:0]` registers.
- **Coordinates:** Scaled by 2^{13} (Q2.13 format).
  Integer = round(Real_Value × 2^{13})
- **Angles:** Scaled by mapping 180° ⟶ 2^{15} (32768).
  Integer = round(Angle_Degrees / 180 × 2^{15})
- **Gain Correction:**
  scaling_factor = 4974 (≈ 0.607 × 2^{13}) applied to initial x_0.
- **Architecture:** Sequential, iterative design controlled by a Finite State Machine (FSM).
- **LUT:** Implemented in `angle_lut.v` using the 2^{15} angle scaling.
- **Parameters:** WIDTH and scaling_factor provide flexibility.

3.2 Finite State Machine (FSM) Design

A 3-state **Moore FSM** controls the operation, using the standard three-block Verilog structure.

States:

- **IDLE:** Waits for `start`. On start, captures inputs (angle ⟶ angle_reg, scaling_factor ⟶ x_reg, 0 ⟶ y_reg), resets `iter_count`.
- **CALC:** Performs one CORDIC iteration per clock cycle for 16 cycles, updating `x_reg`, `y_reg`, `angle_reg`, and incrementing `iter_count`.
- **DONE:** Reached when `iter_count == WIDTH`. Assigns `x_reg` to `cos_out`, `y_reg` to `sin_out`, asserts `done`, then transitions back to IDLE.

4. Simulation and Verification

Functional verification was performed using the **Makerchip IDE**.

The integrated testbench (`cordic_makechip_code.v`) included:

- Clock and reset generation

- Input angles (e.g., 30° scaled to 5461)
- Start pulse
- Module instantiation

Process:
Known input angles were applied, the `start` signal pulsed, and the `done` signal monitored.
Final `cos_out` and `sin_out` values were compared to expected results.

Results:
Simulation confirmed logical correctness.
The FSM operated as designed, and outputs matched expected sine and cosine values within CORDIC's fixed-point error range.
Waveforms (`cordic_makerchip_waveform.vcd`) were captured for verification.

---

5. Conclusion

A 16-bit CORDIC circuit was successfully designed in Verilog with an FSM controller.
Functional verification in Makerchip confirmed adherence to the algorithm and correct fixed-point scaling.

Physical implementation on the IHP SG13G2 PDK was not completed, but the project demonstrates a functional and hardware-efficient trigonometric computation block suitable for larger digital signal processing systems.

Github link: https://github.com/NavyStudent2893/16-bit-CORDIC-Circuit