CORDIC-Based Trigonometric Processor - LogicForge Project Report

1. Introduction

This report explains the design and implementation of a CORDIC (Coordinate Rotation Digital Computer) based system for computing trigonometric functions. The project was developed as part of the LogicForge Digital Design Challenge. We employed a 32-bit fixed-point arithmetic format (Q16.15) to ensure sufficient precision and range for handling full rotations from -π to +π.

2. Why 32-bit Fixed-Point (Q16.15)?

CORDIC algorithms compute angles typically in the range [-π, +π]. To accurately represent these angles in fixed-point, we use Q16.15 format, which allocates 16 bits for the integer part (including sign) and 15 bits for the fractional part. This allows the system to represent values between approximately -32768.0 to +32767.999969. In this format, π ≈ 3.1416 is encoded as 102944 (π × 32768), which would overflow in a 16-bit signed integer format such as Q1.15. Hence, a 32-bit representation is mandatory for correctness and stability of trigonometric operations.

3. How CORDIC Works

The CORDIC algorithm is an iterative method that rotates vectors to compute trigonometric, hyperbolic, and linear functions. It works entirely with shifts, adds, and subtracts —making it highly efficient in hardware. In rotation mode, it rotates a vector by a given angle. In vectoring mode, it computes the magnitude and angle of a vector.

Rotations in the Plane-Simplifications

$$x' = x\cos\alpha - y\sin\alpha$$
$$y' = y\cos\alpha + x\sin\alpha$$

Factoring the term cosα from the above Equations:

$$x' = \cos\alpha \cdot [x - y\tan\alpha]$$
$$y' = \cos\alpha \cdot [y + x\tan\alpha]$$

Restrict the Rotation Angles Such that:

$$\tan\alpha = \pm 2^{-i}$$

Above Multiplication is Reduced to a Simple Shift!!

The basic iterative formula is:

$$x_{(i+1)} = x_i + y_i * d_i * 2^{(-i)} \quad *$$
$$y_{(i+1)} = y_i + x_i * d_i * 2^{(-i)}$$
$$z_{(i+1)} = z_i - d_i * \arctan\left(2^{(-i)}\right)$$

Where d_i is the direction (+1 or -1), and the iteration count defines the precision.

4. FSM and Control Logic

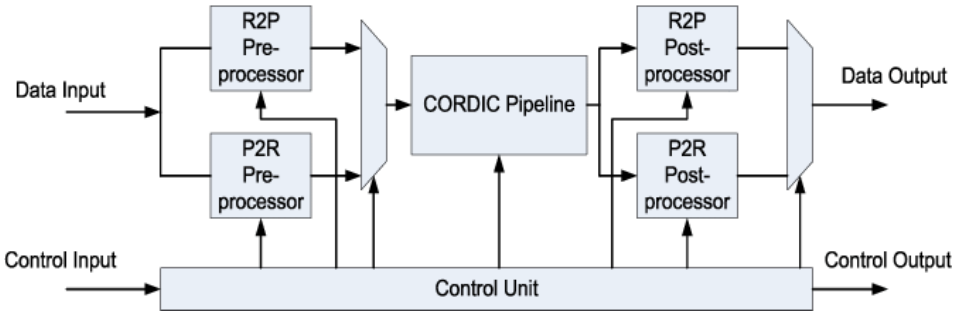
The CORDIC system is implemented using a Finite State Machine (FSM) with three states: IDLE, PROCESS, and DONE. In the IDLE state, inputs are loaded. In the PROCESS state, iterative calculations are performed over 16 cycles. The DONE state signals the availability of results. The FSM enables sequential control of the pipeline, ensuring correct timing and synchronization for CORDIC operations.

5. Implementation Notes

• The atan lookup table uses fixed-point Q16.15 format and is stored as a 32-bit array.
• The gain factor K ≈ 0.60725 is pre-applied in rotation mode to avoid multiplication at the output.
• All input/output signals are 32-bit to accommodate Q16.15 range.
• Testbench verifies both rotation and vectoring modes with π/4 and atan(1) respectively.

6. Block Diagram and FSM

The diagram below illustrates the high-level block design of the CORDIC processor along with its FSM (Finite State Machine). The system transitions from IDLE to PROCESS state for computation and then to DONE when iterations are complete.



[FSM (*CONTROL UNIT*): IDLE → PROCESS → DONE]

7. Simulation Results

The system was tested using a Verilog testbench for two use cases:

- Rotation mode with input angle π/4 (expected sin ≈ cos ≈ 0.7071)
- Vectoring mode with x = y = 0.7071 (expected angle ≈ π/4)
- Since we are working in Fixed Point Representation in Q16.15 so the input must be given in Q16.15 format.

Sample Input

Rotation mode:

- Angle = π/4 (0.78540)
- Q16.15 representation of π => 3.14159 * 2^{15} = 102944 (PI_FIXED)
- Q16.15 representation of π/4 = (PI_FIXED)>>>2

Vectoring mode:

- x = y = 0.7071
- Q16.15 representation = 0.7071 * 2^{15} = 23170

Sample Output

Rotation mode:

- Cos = 0.707123 , Sine = 0.707092

Vectoring mode:

- Magnitude = 1.000012
- Angle in radian = 0.785492 ≈ π/4

8. Conclusion

The 32-bit Q16.15 CORDIC design offers a robust and hardware-efficient solution for computing trigonometric functions. It provides sufficient precision and range for angles in the full [-π, π] domain, making it suitable for embedded, DSP, and FPGA-based applications.