



## **Assignment 2**

### **Design of CORDIC algorithm using Verilog**

Submitted By:

Rahul	2023EEN2238
Sahil Dhiman	2023EEN2656
Adarsh Mittal	2023EEN2234
Sanket Chaudhary	2023EEN2231

# 1 Introduction

Coordinate Rotation Digital Computer (CORDIC) is a simple and efficient algorithm to compute arithmetic, trigonometric and hyperbolic functions. CORDIC algorithm is an iterative algorithm which evaluates a function by successive clockwise or anticlockwise micro rotations of coordinates. The benefit of the CORDIC algorithm is that the microrotations are calculated by simple shift-and-add operations, which is very efficient in hardware.

In this assignment, sine and cosine of the angle is calculated using the CORDIC algorithm. The range of the angles covered is from  $\pi/2$  radians and  $(-\pi/2)$  radians. The input angle is 8-bit number with 1 sign bit, 1 integer bit and 6 fractional bits. The outputs are also 8-bit number. The Verilog code for calculating the sine and cosine of the angle using CORDIC algorithm is simulated in the ModelSim simulation environment. The design is then synthesized using the Cadence Genus tool.

## 1.1 Referenced documents

- M. Garrido, P. Källström, M. Kumm and O. Gustafsson, "**CORDIC II: A New Improved CORDIC Algorithm**," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 2, pp. 186-190, Feb. 2016, doi: 10.1109/TCSII.2015.2483422.
- Duprat, Jean, and J-M. Muller. "**The CORDIC algorithm: New results for fast VLSI implementation**." *IEEE transactions on computers* 42.2 (1993): 168-178.
- Hu, Xiaobo, Ronald G. Harber, and Steven C. Bass. "**Expanding the range of convergence of the CORDIC algorithm**." *IEEE Transactions on computers* 40.01 (1991): 13-21.

## 1.2 Design library name

UMC-65 nm

## 1.3 People involved in the block

- Sahil Dhiman
- Sanket Chaudhary
- Adarsh Mittal
- Rahul

# 2 Function

## 2.1 Brief overview

The brief overview of the CORDIC algorithm is given in this section.

Consider a vector at its initial position  $[x, y]$ . Rotating a vector from its initial position by an angle  $\theta$  results in a vector at  $[x_k, y_k]$  in the cartesian plane which is given by :

$$\begin{aligned}x_k &= x\cos\theta - y\sin\theta \\y_k &= x\sin\theta + y\cos\theta\end{aligned}$$

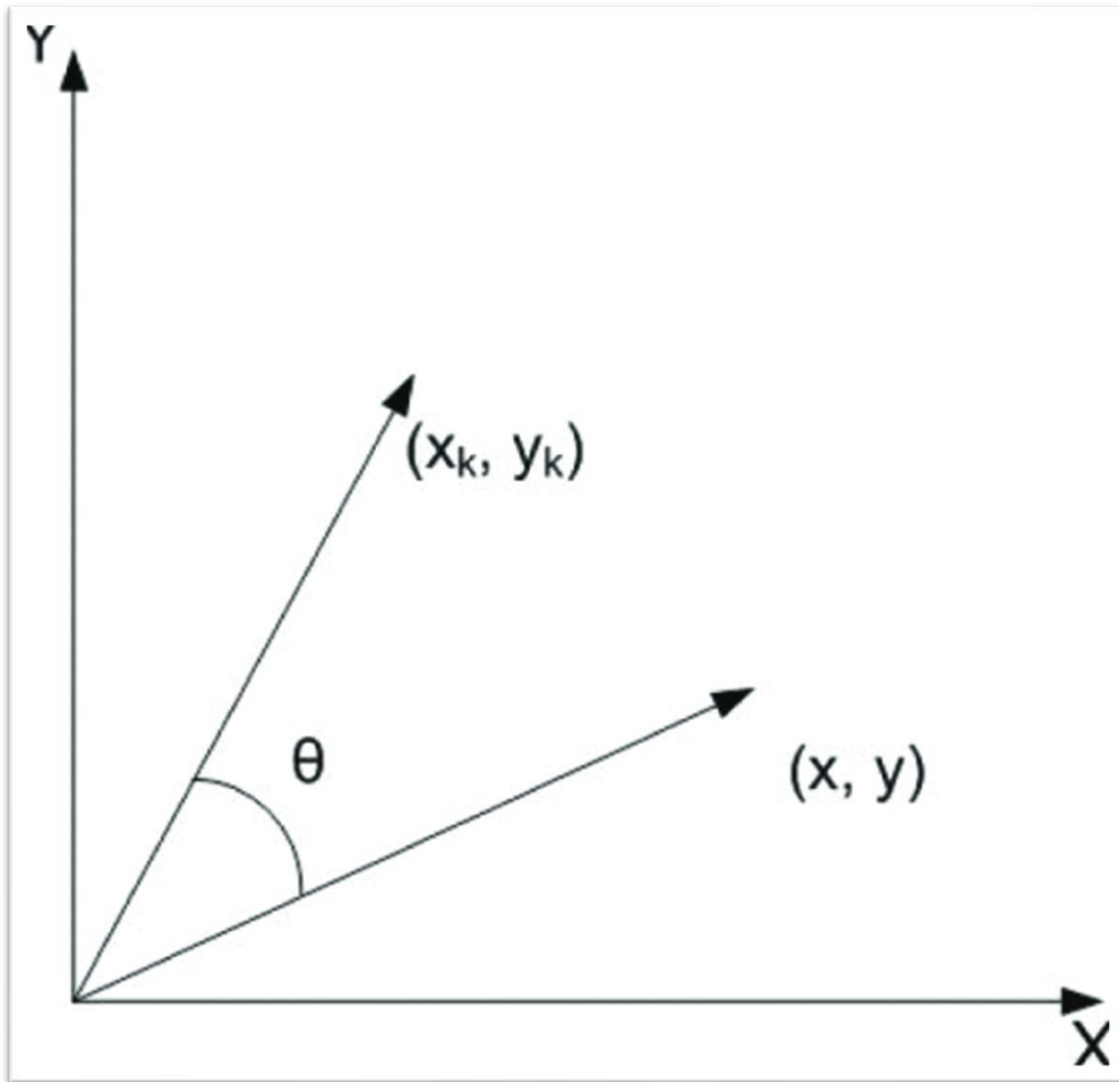


Figure 1 Cartesian Coordinates Representation of vector

Factoring out the  $\cos\theta$  term,

$$\begin{aligned}x_k &= \cos\theta(x - y\tan\theta) \\y_k &= \cos\theta(x\tan\theta + y)\end{aligned}$$

The required angle of rotation  $\theta$  is obtained by performing a series of small elementary rotations ( $\phi$ ). At each iteration, these small rotational angles ( $\phi$ ) decrease the error ( $z$ ) between the desired and obtained angle. Mathematically,  $\theta$  and  $\phi$  are represented as:

$$\begin{aligned}\theta &= \sum_{i=0}^{n-1} d_i \phi_i \\ \tan \phi_i &= \pm 2^{-i}\end{aligned}$$

Variable  $d_i$  indicates the direction of rotation which reduces error angle. Anti-clockwise and clockwise rotations are denoted with value +1 and -1 respectively. The desired angle of rotation  $\theta$  can be expressed as a function of number of micro-rotations ( $n$ ) given by

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \prod_{i=0}^{n-1} \cos \phi_i \begin{bmatrix} 1 & -d_i 2^{-i} \\ d_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

At each iteration  $i$ , small rotational angles are accumulated and the error angle ( $z$ ) gets updated. The error angle ( $z$ ) can now be expressed as a function of  $i^{\text{th}}$  iteration, direction of rotation  $d_i$  and initial vector position ( $x_i, y_i$ ) given by:

$$d_i = \pm 1$$

$$K_i = \prod_{i=0}^{n-1} \cos(\tan^{-1}(2^{-i})) = 1 / \left( \prod_i \sqrt{1 + 2^{-2i}} \right)$$

As the number of iteration increases, product of  $\cos \phi_i$  approaches 0.6074.  $K_i$  is treated as a gain factor for this algorithm and attains a value of 1.647. Removing the scale constant  $K_i$ , solves the vector rotation problem by shift and add operations.

The micro-rotational angles ( $\phi$ ) are precomputed and stored in a look-up table. Accumulating these micro-rotational angles limits the range between  $\pi/2$  radians and  $(-\pi/2)$  radians. So, conventional CORDIC can be operated in the 1<sup>st</sup> and 4<sup>th</sup> quadrant.

## 2.2 Interfaces

Signal name	I/O	Description
in_angle [7:0]	INPUT	8-bit Input angle
clk	INPUT	Clock signal
rst	INPUT	Reset signal
x_cosine [7:0]	OUTPUT	8-bit cosine of input angle
y_sine [7:0]	OUTPUT	8-bit sine of input angle

Table 1 Description of I/Os used in the CORDIC algorithm.

## 2.3 Architecture

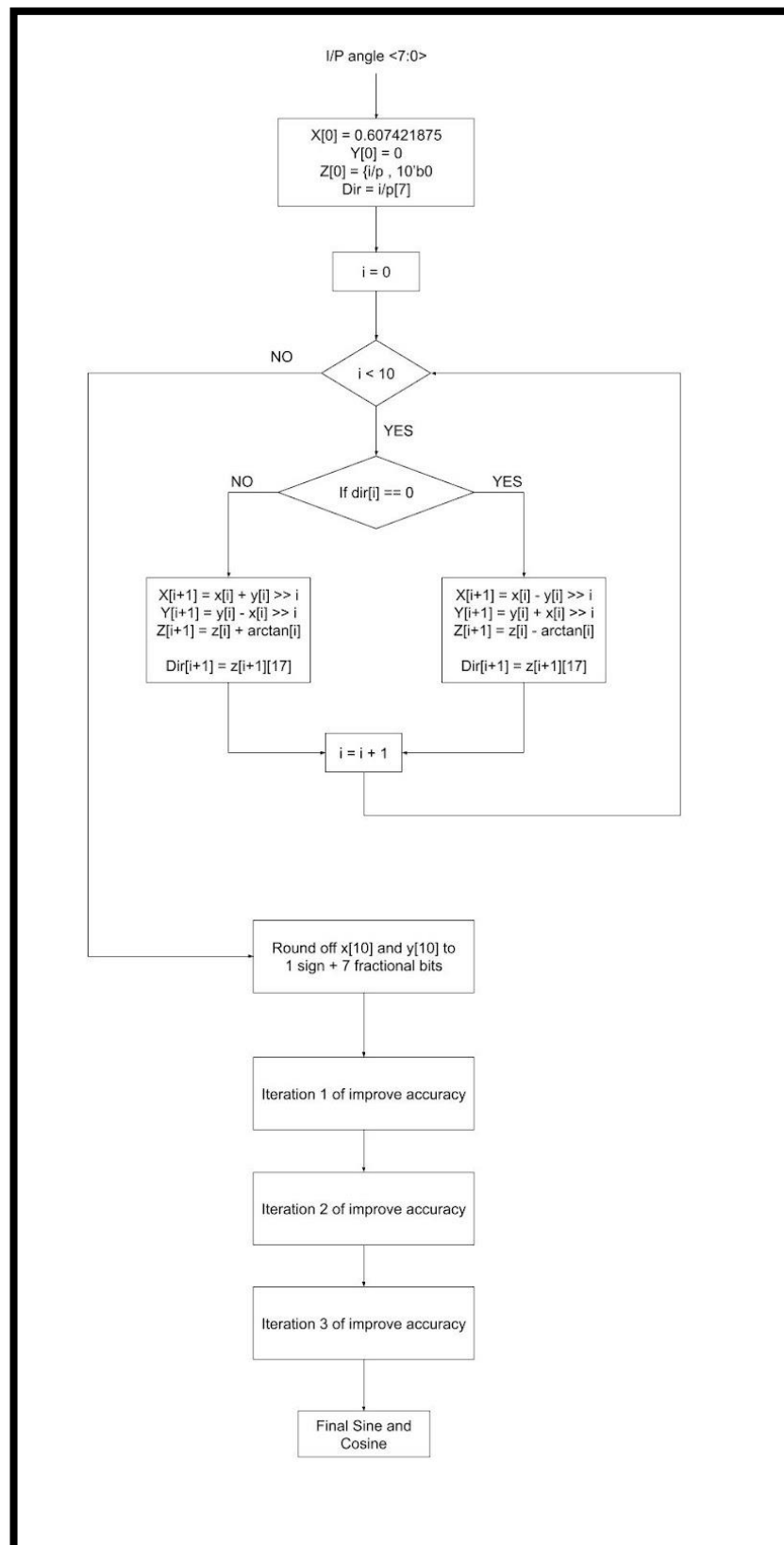


Figure 2 Architecture of CORDIC algorithm

## **2.4 Detailed functional description**

- The input to the CORDIC algorithm design is 8-bit input angle in radians.
- It contains 1 sign bit, 1 integer bit and 6 fractional bits.
- The output is sine and cosine of the input angle with 1% error.
- To improve the accuracy, 8 bit input is made as 18 bit by padding zeroes. The output after 10 iterations of CORDIC algorithm is of 18-bit with 1 sign bit, 1 integer bit and 16 fractional bits.
- This 18-bit output is highly accurate but when we round off to 8 bits for final output, accuracy degrades significantly. Thus, to improve the accuracy after rounding this module is used.
- Round-off circuit used to round-off the 16 fractional bits of the output to 7 bits.
- The basic idea is to calculate the error between 18-bit CORDIC output and the rounded off output. to determine if we need to add or subtract 1 at LSB to reduce the error.
- If error is already small enough, no changes are made to the output.
- To determine if error is big enough or not, the first 10 bits from MSB are checked. If any one of them is 1, we need to add/subtract 1 to/from LSB otherwise not.
- Since the rounded off output is of only 8 bits, we need to sign extend by 1 bit and add 0's after LSB to make it 18 bits so that error can be calculated.
- The Verilog code is simulated on ModelSim simulation environment.
- Cadence genus tool is used to synthesise the design.

## **3 Design parameters**

### **3.1 Performance Requirements**

- To achieve a max frequency of 100 MHz.
- To achieve less than 1 % error in the outputs.

### **3.2 Clock Distribution**

- The frequency of the clock signal is 100 MHz.
- The duty cycle of the clock is 50%.
- Rise and fall time of clock is 0.1 ns.
- Active clock edge is Positive edge of the clock.

### **3.3 Reset**

- The reset signal is asynchronous and is active low.

### 3.4 Timing Description

- Latency: 150ns (15 Clock Cycles)
- Update conditions: At every positive edge of clock.

## 4 Verification Strategy

### 4.1 Objectives

- To keep the error in sine and cosine of the input angle less than 1%.
- Design should operate at a maximum frequency of 100 MHz.

### 4.2 Tools and Version

- Synthesis - Genus
- Place and Route - Innovus
- Simulation - ModelSim

### 4.3 Checking mechanisms

- Input angle is swept from -1.5625 radians (-89.52°) to + 1.5625 radians (+89.52°).
- For some specific input angles, the sine and cosine of input angle are tabulated below.

Table 2 Simulation results on ModelSim

Input Angle (degrees)	Cosine of input angle	Sine of input angle
-80.572190	0.164063	-0.984375
-63.562505	0.445313	-0.898438
-47.448067	0.679688	-0.734375
-34.914616	0.820313	-0.570313
-26.857397	0.890625	-0.453125
-20.590671	0.937500	-0.351563
40.286095	0.765625	0.648438
55.505286	0.570313	0.820313



67.143492	0.390625	0.921875
74.305464	0.273438	0.960938

- The simulation output data of ModelSim was used to compute the error. The figure 2 shows the error in sine and cosine of input angles.

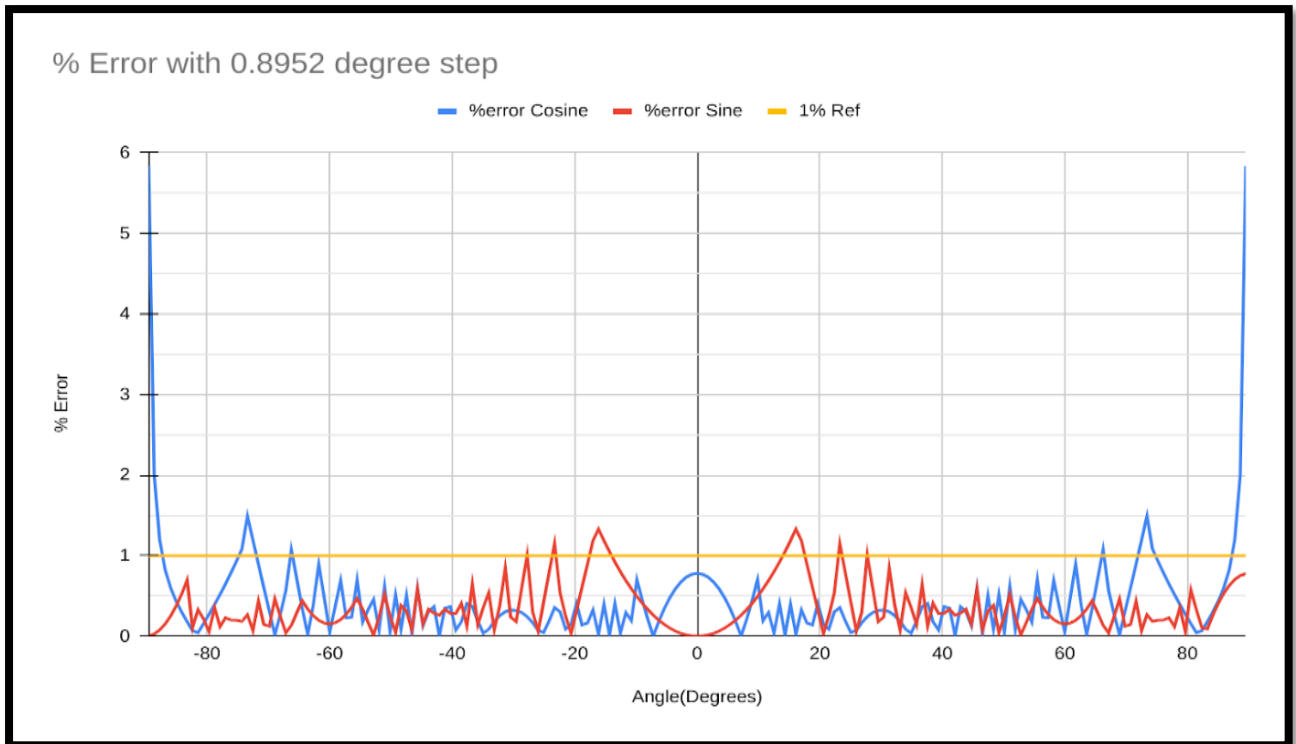


Figure 3 % error in sine and cosine of input angle

## 5 Functional Checklist

The following things needs to be checked:

1. Operating Frequency: 100MHz
2. Setup slack is at least 30% of clock time period.
3. Error in cosine and sine for all inputs is less than 1%.

## 5 Testbench

### 5.1 Overview

- The testbench initially sends the reset signal to reset all registers to 0.

- The input angle is swept from -1.5625radians (-89.52 degrees) to + 1.5625 radians (+89.52 degrees).
- The output is logged in console in csv format which is then later used to compute accuracy to plot a graph.

## 5.2 Architecture

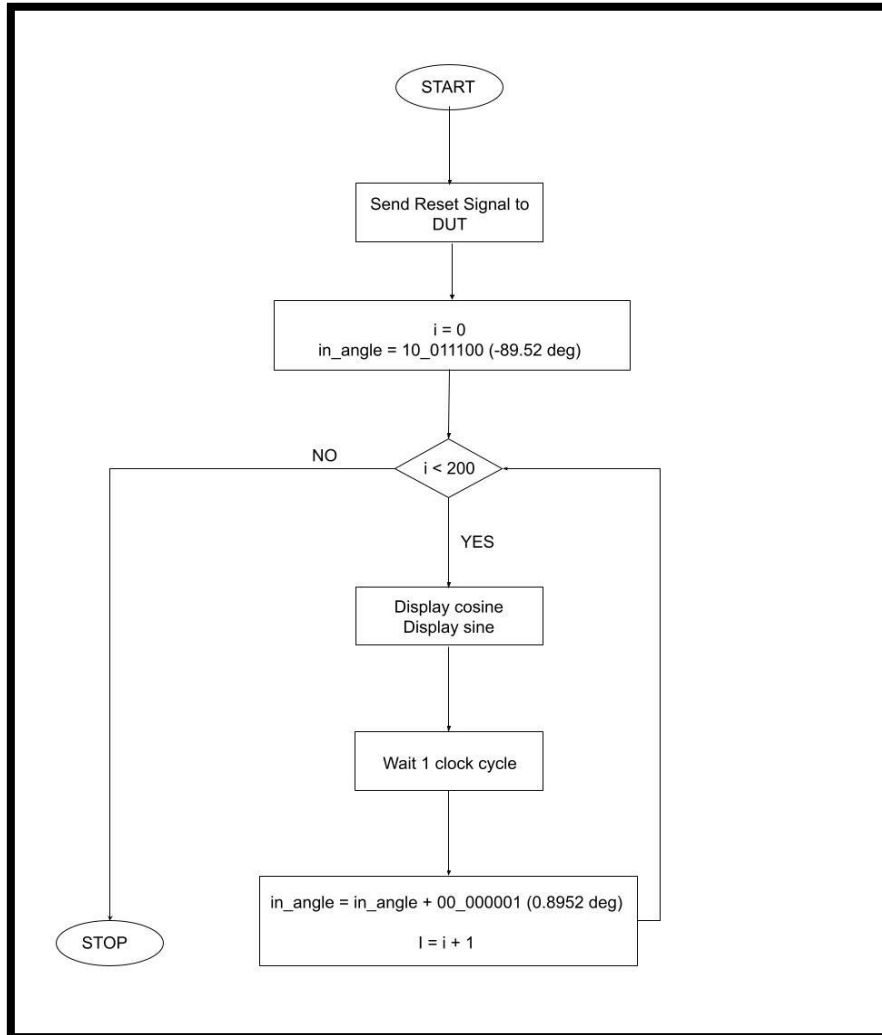


Figure 4 Architecture of the Testbench

## 6 Tests Specification

- Input angle is swept from -1.5625 radians (-89.52 degrees) to + 1.5625(+89.52 degrees).
- The output is compared to actual sine and cosine values and graph is plotted using google sheets.

The following Verilog code is used in the testbench to achieve this:

```

for(j=0; j<200; j=j+1)
begin
if(j<1)
in_angle = 8'b10_011100;
else
in_angle = in_angle + 8'b00_000001;
end

```

## 7 Design Microarchitecture

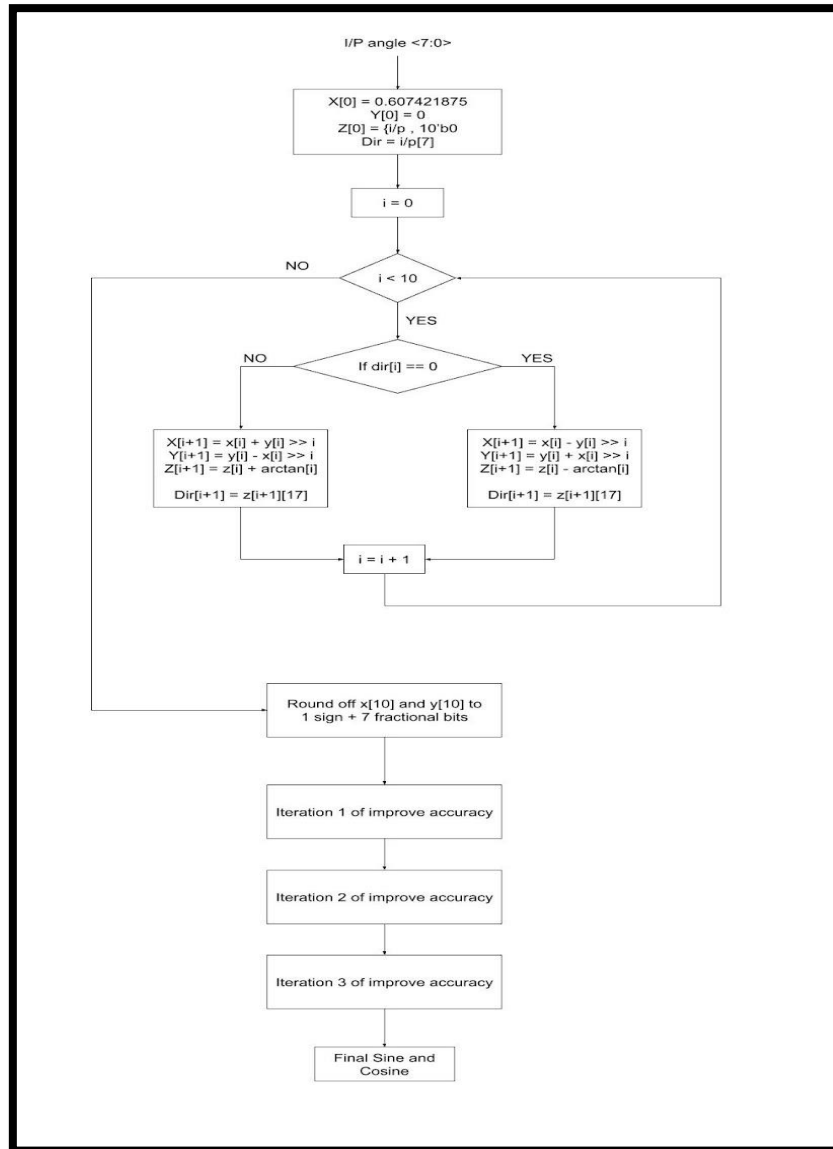


Figure 5 Design Microarchitecture

### 7.1 Top Level Interface

NA

## 7.2 Sub-Block Description

- Sub-blocks used in the overall design:
  1. Improve Accuracy
  2. Round Off
  3. Mux 2 to 1

### 1. Improve Accuracy

- This module is responsible for improving the accuracy of sine and cosine of the input angle calculated after 10 iterations of algorithm.
- Since before starting the CORDIC algorithm, the input angle is extended to 18 bits by adding 10 zeros at LSB, the output after 10 iterations of CORDIC, we get 18-bit output with 1 sign bit, 1 integer bit and 16 fractional bits.
- This 18-bit output is highly accurate but when we round off to 8 bits for final output, accuracy degrades significantly. Thus, to improve the accuracy after rounding this module is used.

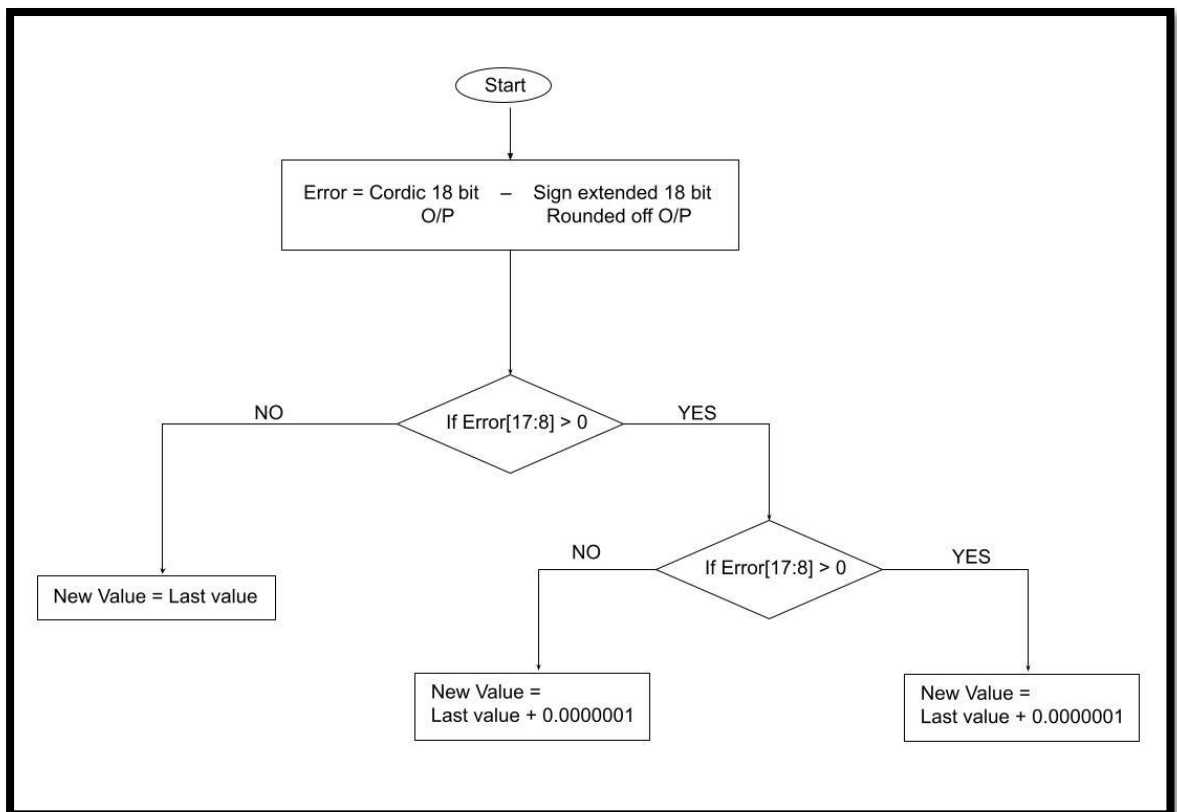


Figure 6 Flowchart of Improve accuracy block

- The basic idea is to calculate the error between 18-bit CORDIC output and the rounded off output. to determine if we need to add or subtract 1 at LSB to reduce the error.
- If error is already small enough, no changes are made to the output.
- To determine if error is big enough or not, the first 10 bits from MSB are checked. If any one of them is 1, we need to add/subtract 1 to/from LSB otherwise not.
- Since the rounded off output is of only 8 bits, we need to sign extend by 1 bit and add 0's after LSB to make it 18 bits so that error can be calculated.

## 2. Round Off

Round-off circuit used to round-off the 16 fractional bits of the output to 7 bits of fractional part.

- If the in[8] bit is equal to 0 then the number is rounded down.
- If the in[8] bit is equal to 1 and there is at least one 1 in the following bits then the number is rounded up.
- If the in[8] bit is equal to 1 and all the following bits are 0, then it is condition of tie break which is resolved by even-ties
- In the last case, if in[9] is 0 then the number is rounded down and if it is 1 then the number is rounded up.

## 3. Mux 2 to 1

This module is a 2:1 Multiplexer.

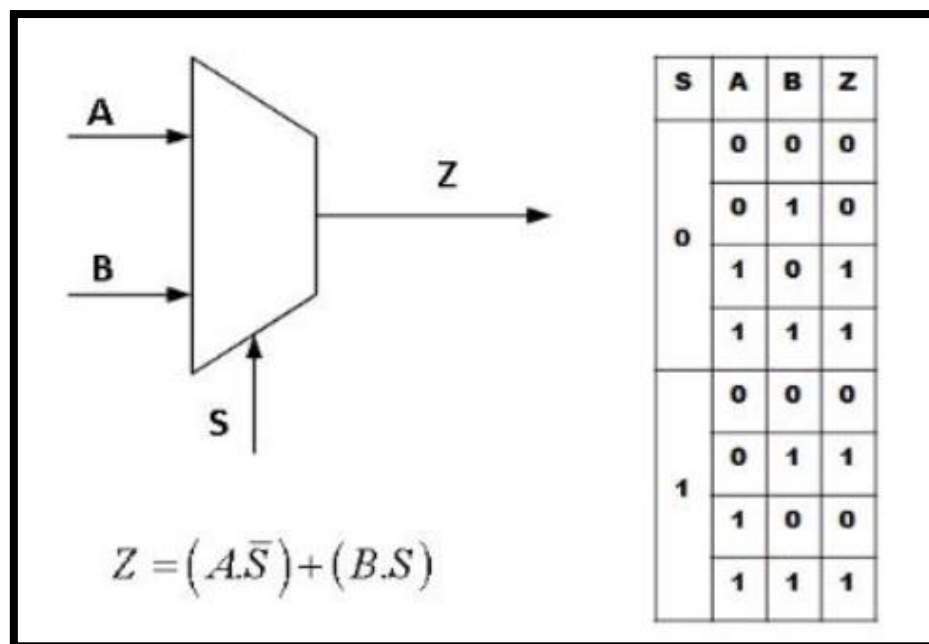


Figure 7 2:1 MUX

### 7.3 Structural Mapping Process

- The RTL was converted into the gate-level netlist using the logic synthesis in the Genus tool.
- The generated netlist was further used for place and route using the innovus tool.

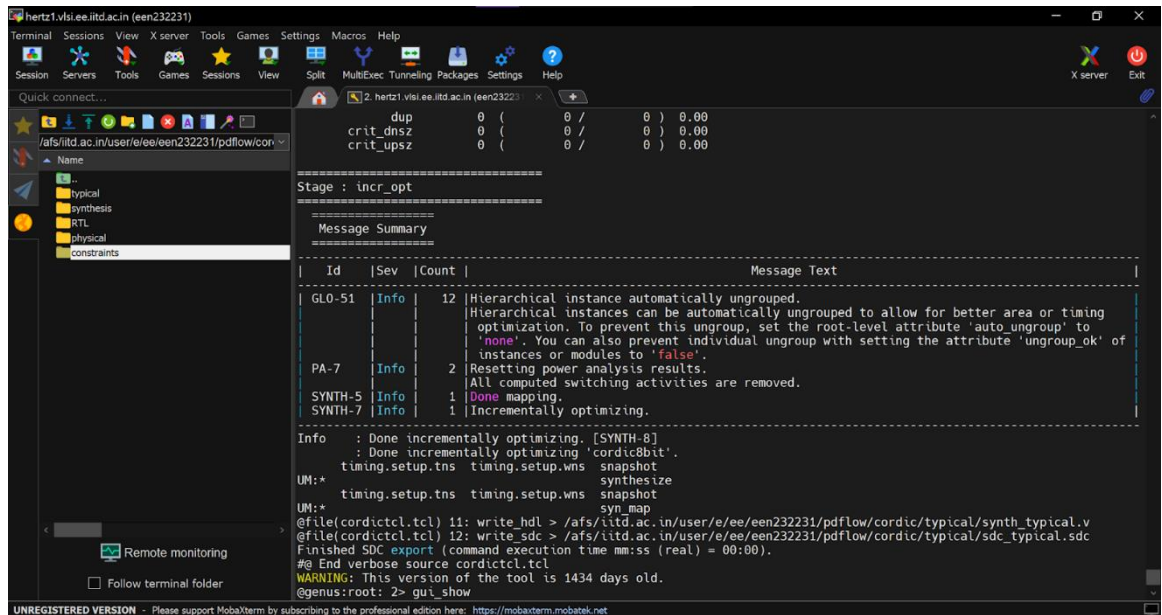


Figure 8 Netlist generation from RTL

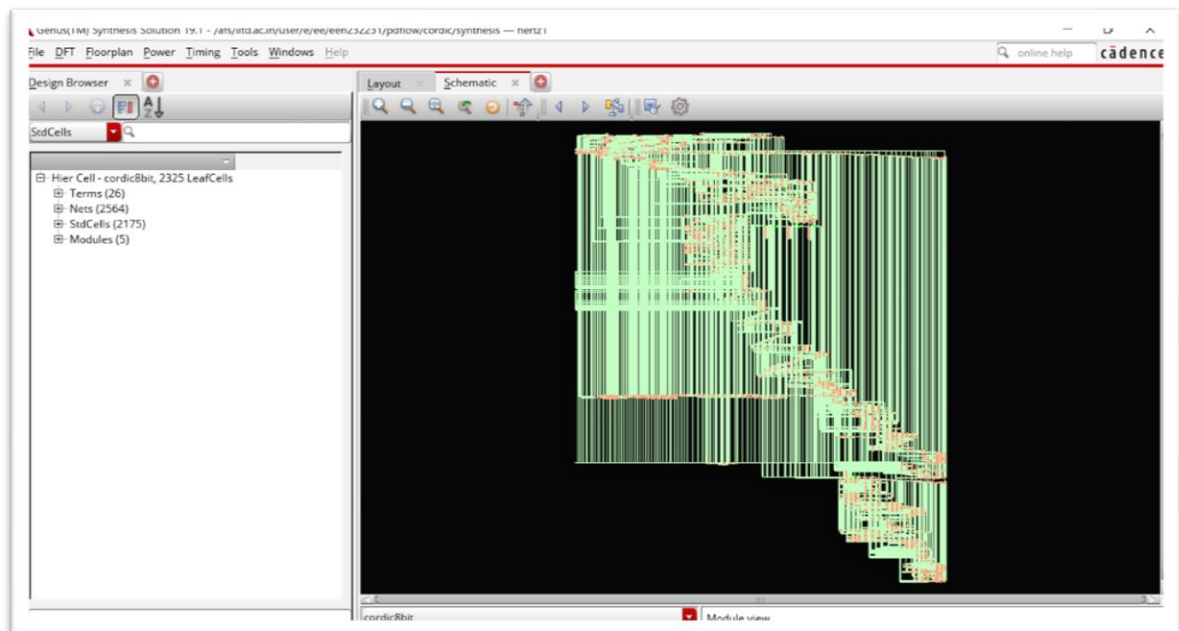


Figure 9 Schematic of CORDIC algorithm design

## 8 Physical hierarchy

### 8.1 Floor planning

Floorplan was specified at the beginning step with the following configurations:

- Aspect Ratio was taken as 1
- Core Utilization was taken as 0.7
- Core margins were selected as Core\_to\_Boundary with a value of 4.5 on each side.

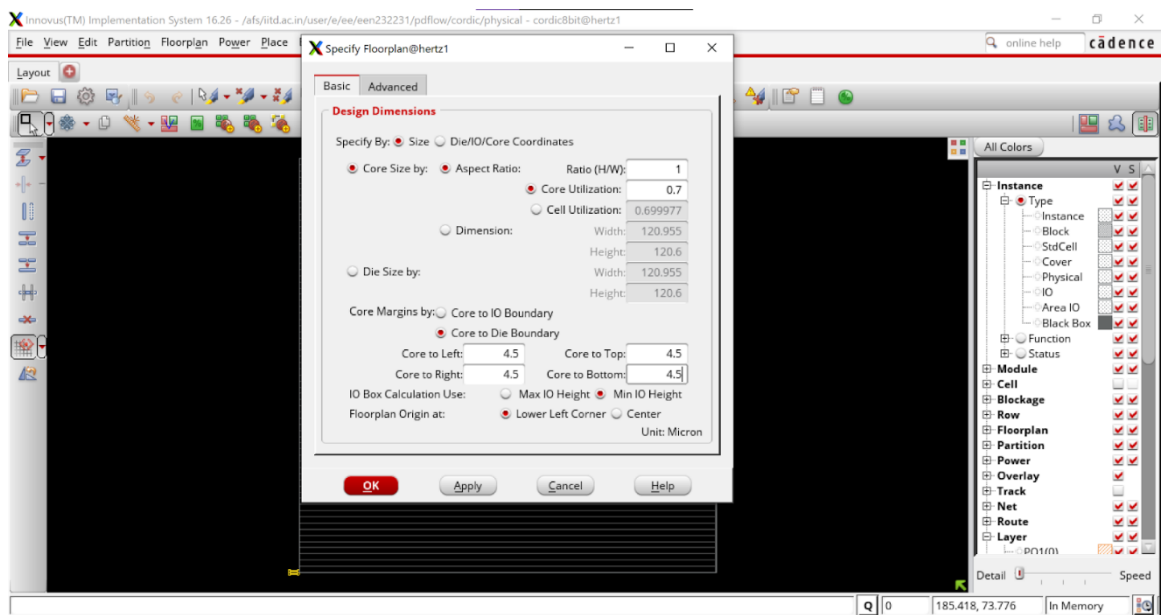


Figure 10 Floorplan

### 8.2 Clocktree insertion

Clock constraints were provided to the synthesis tool:

1. `create_clock -name "clk" -add -period 6.66 -waveform {0 3.33} [get_ports "clk"]`
2. `set_clock_transition -rise 0.66 [get_clocks "clk"]`
3. `set_clock_transition -fall 0.66 [get_clocks "clk"]`
4. `set_clock_uncertainty 0.066 [get_clocks "clk"]`
5. `set_clock_latency 0 [get_ports "clk"]`

### 8.3 Layout Strategy

- From the generated netlist, firstly the innovus tool was invoked for layout generation.
- All the MMC view configurations were defined, such as the maximum and minimum timing libraries, default RC corner, maximum and minimum delay corners, constraints applied and finally setting up the analyses view for best case and worst case situations.
- After specifying the floorplan, the power planning was performed i.e, addition of ring and stripes.
- ME8 and ME7 were used for adding VDD and VSS rings and ME6 was used to add stripes.
- This was followed by special route, placement of standard cells, clock tree synthesis and nano routing.

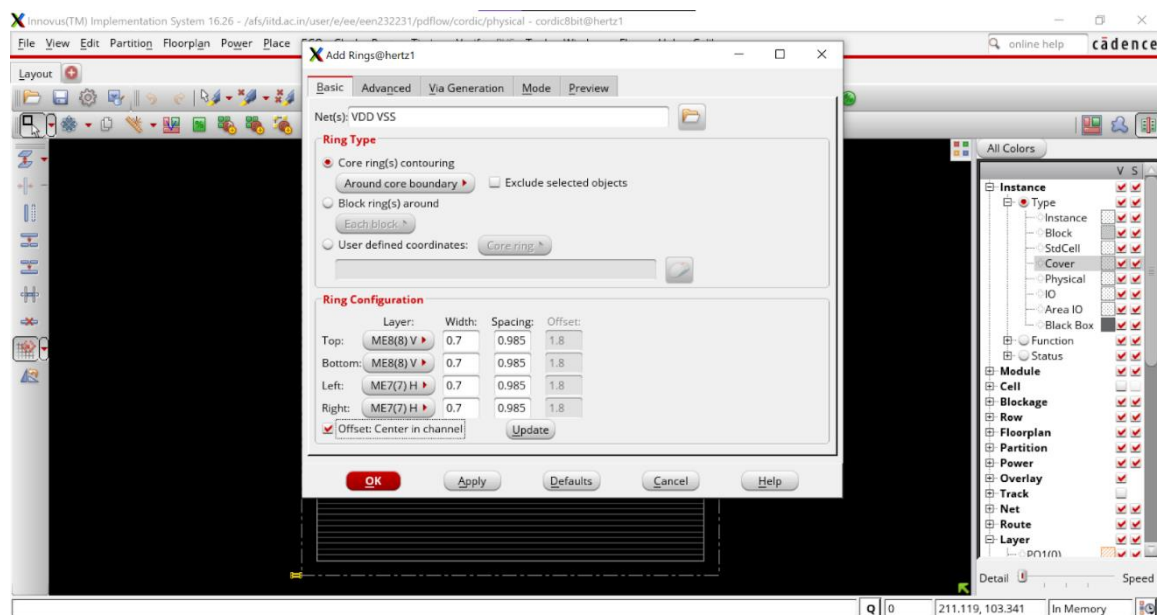


Figure 11 Addition of VDD and VSS rings

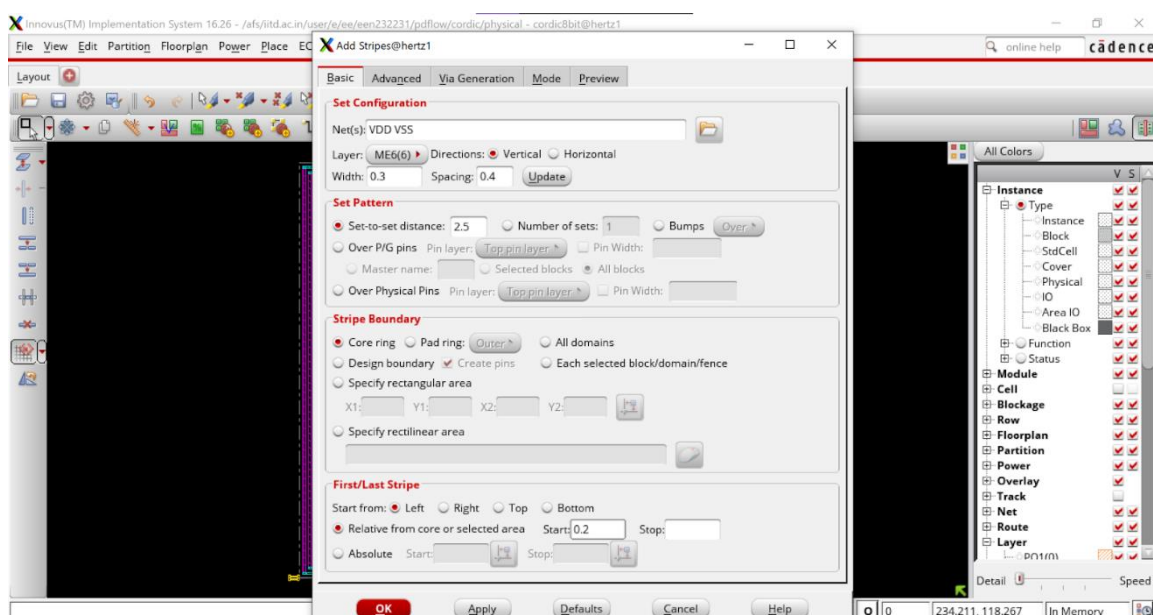


Figure 12 Addition of stripes



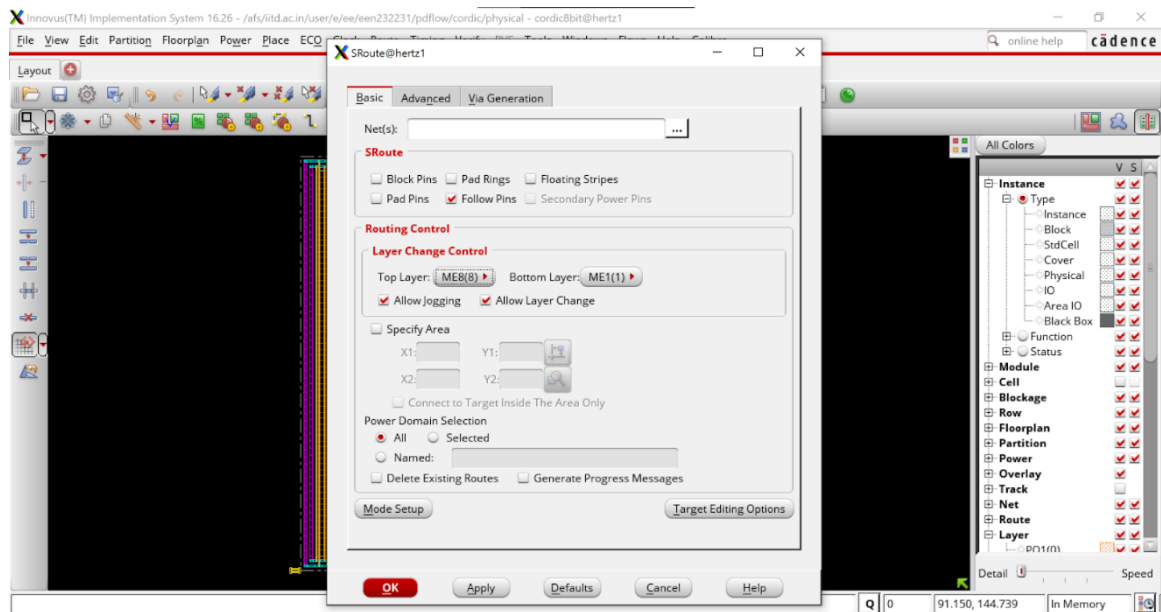


Figure 13 Special route

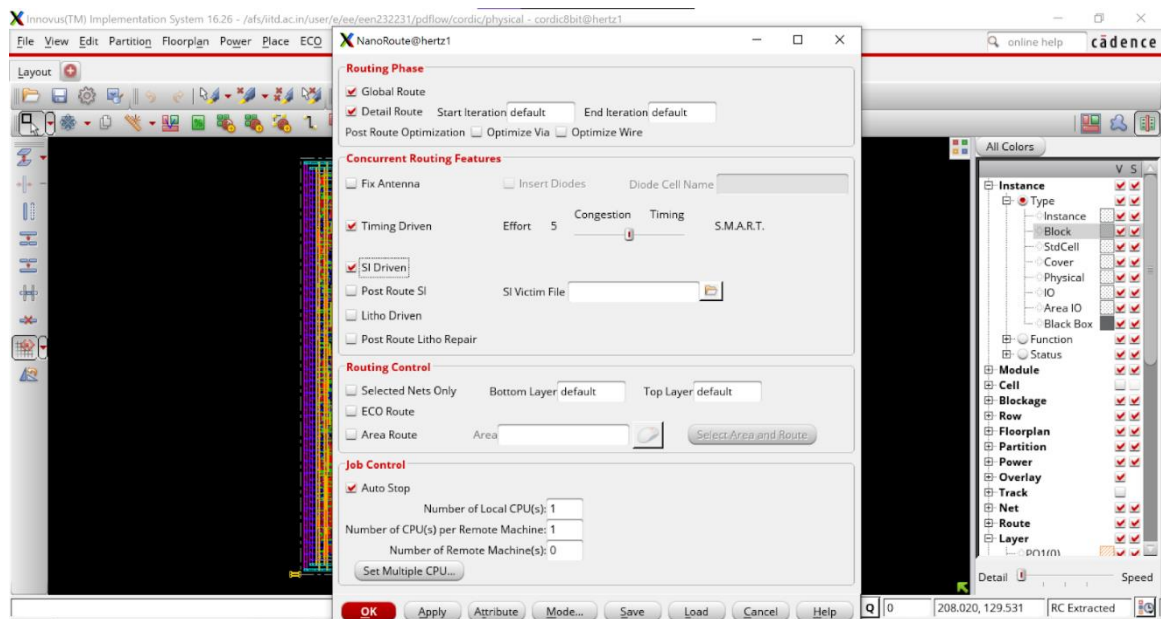


Figure 14 Nano Route

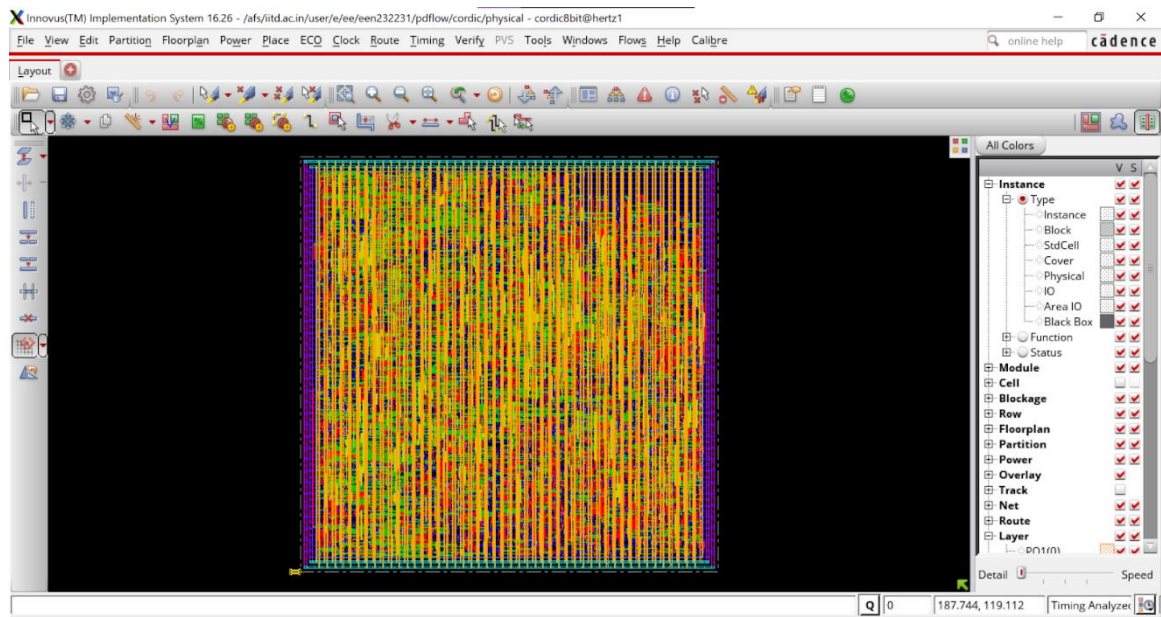


Figure 15 Final Layout

## 9 Results

### 9.1 Area

The reported area of both the units is  $10446.48 \mu\text{m}^2$ .

Table 3 Area of CORDIC algorithm design

Block Name	Standard Cells Count	Dimensions (W×L)	Layout Area( $\mu\text{m}^2$ )
CORDIC	2358	$102.21 \times 102.21 \mu\text{m}$	10446.48

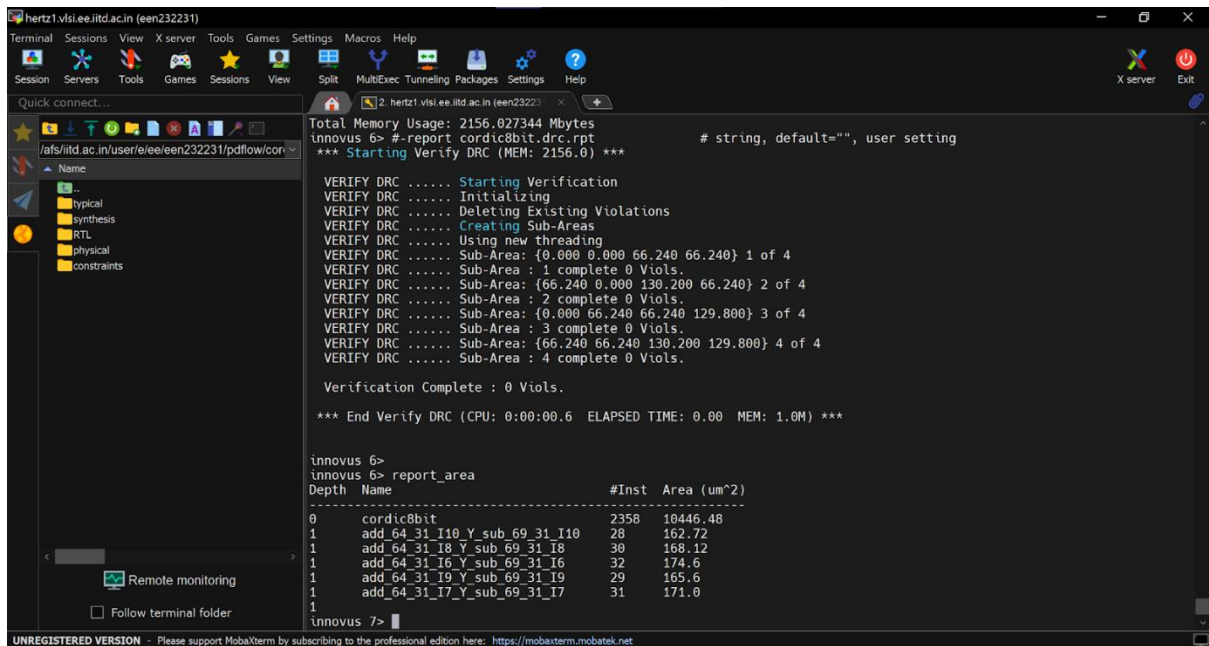


Figure 16 Screenshot of Area reported

## 10.2 Timing

The timing report of both the blocks is as follows:

Table 4 Timing reports

Timing	Setup (ns)	Hold (ns)
Pre-CTS	3.882	0.080
Post-CTS	3.892	0.076

- The clock cycle time period is 10ns and clock skew is not considered in the analysis.

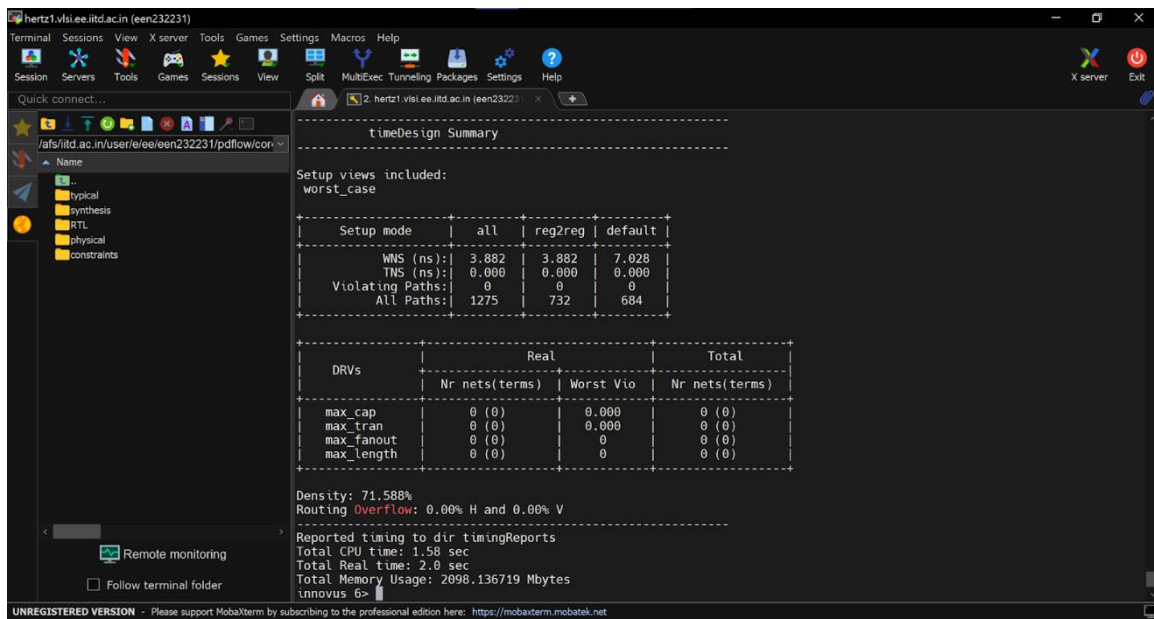


Figure 17 Pre-CTS Setup

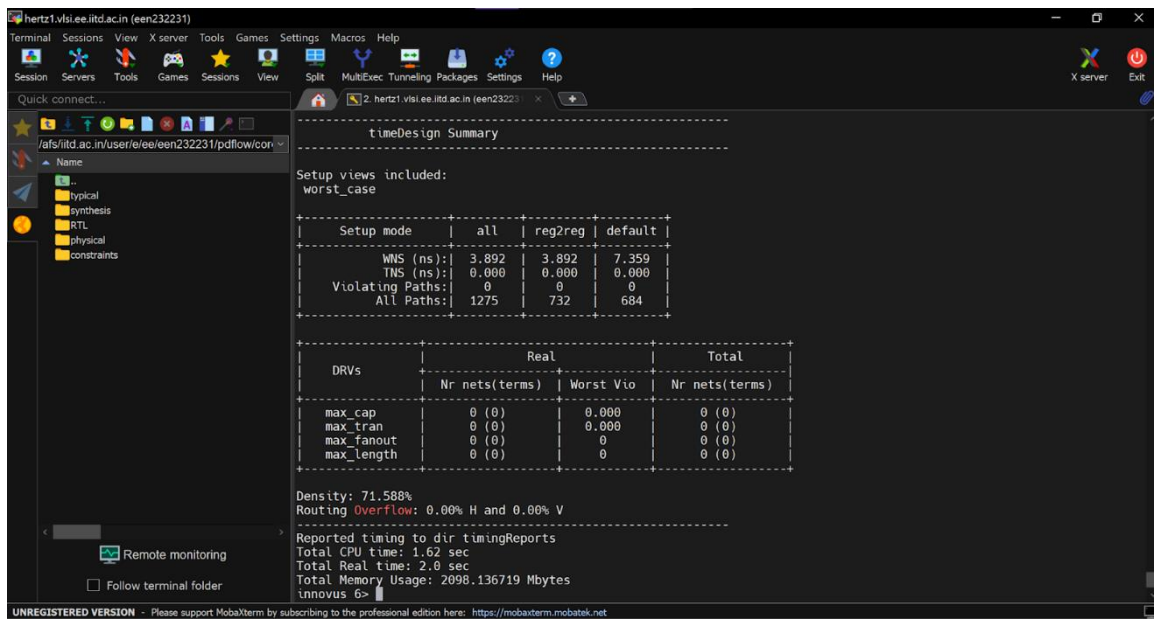


Figure 18 Post-CTS Setup

```

# Signoff Settings: SI Off
#####
AAE_INFO: 1 threads acquired from CTE.
Calculate delays in BcWc mode...
*** Calculating scaling factor for min_timing_library libraries using the default operating condition of each library
.
Total number of fetched objects 2742
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=2154.36 CPU=0:00:00.7 REAL=0:00:01.0)
*** Done Building Timing Graph (cpu=0:00:00.9 real=0:00:01.0 totSessionCpu=0:04:14 mem=2154.4M)

-----
timeDesign Summary
-----

Hold views included:
best_case

-----
| Hold mode | all | reg2reg | default |
-----
| WNS (ns): | 0.080 | 0.080 | 0.000 |
| TNS (ns): | 0.000 | 0.000 | 0.000 |
| Violating Paths: | 0 | 0 | 0 |
| All Paths: | 732 | 732 | 0 |
-----

Density: 71.588%
Routing Overflow: 0.00% H and 0.00% V
-----
Reported timing to dir timingReports
Total CPU time: 1.35 sec
Total Real time: 2.0 sec
Total Memory Usage: 2078.980469 Mbytes
unnovus 6>

```

Figure 19 Pre-CTS Hold

```

# Signoff Settings: SI Off
#####
AAE_INFO: 1 threads acquired from CTE.
Calculate delays in BcWc mode...
*** Calculating scaling factor for min_timing_library libraries using the default operating condition of each library
.
Total number of fetched objects 2742
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=2154.36 CPU=0:00:00.7 REAL=0:00:01.0)
*** Done Building Timing Graph (cpu=0:00:00.9 real=0:00:01.0 totSessionCpu=0:04:18 mem=2154.4M)

-----
timeDesign Summary
-----

Hold views included:
best_case

-----
| Hold mode | all | reg2reg | default |
-----
| WNS (ns): | 0.076 | 0.076 | 0.000 |
| TNS (ns): | 0.000 | 0.000 | 0.000 |
| Violating Paths: | 0 | 0 | 0 |
| All Paths: | 732 | 732 | 0 |
-----

Density: 71.588%
Routing Overflow: 0.00% H and 0.00% V
-----
Reported timing to dir timingReports
Total CPU time: 1.38 sec
Total Real time: 1.0 sec
Total Memory Usage: 2078.980469 Mbytes
unnovus 6>

```

Figure 20 Post-CTS Hold



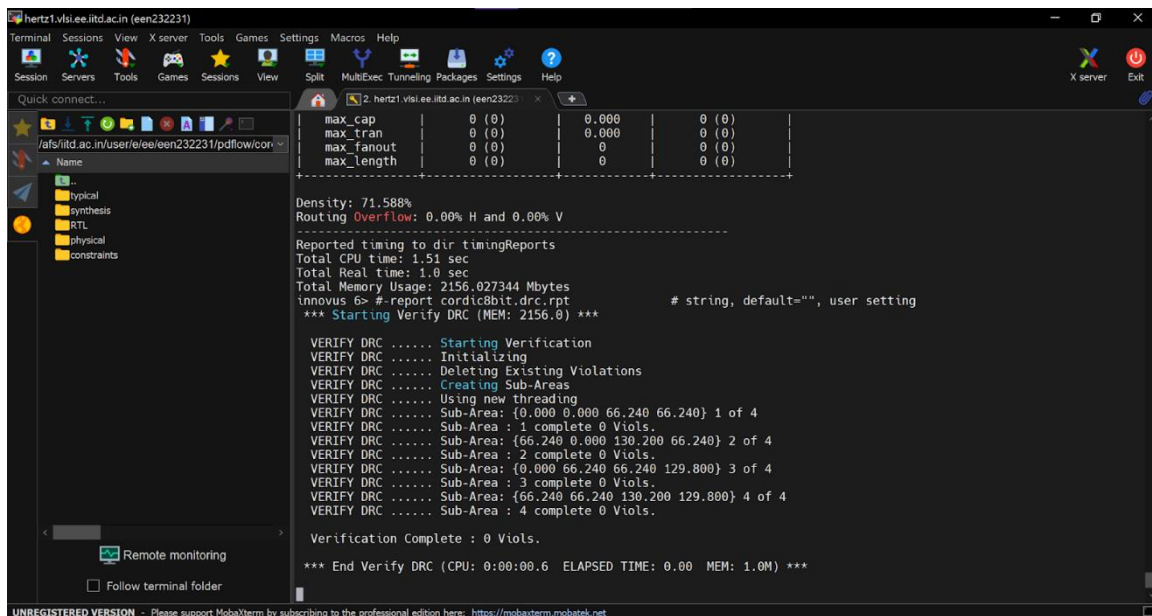
## 10.3 Testability analysis

### Test signals

- clk – Input clock signal
- rst – reset signal
- in\_angle – input angle
- x\_cosine – cosine of input angle
- y\_sine – sine of input angle

## 10.4 DRC rule violations

- No DRC rule violations were reported.



```
hertz1.vlsi.ee.iitd.ac.in (een232231)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
/afs/iitd.ac.in/user/e/een232231/pdf/flow/con...
Name
typical
synthesis
RTL
physical
constraints
max_cap 0 (0) 0.000 0 (0)
max_tran 0 (0) 0.000 0 (0)
max_fanout 0 (0) 0 0 (0)
max_length 0 (0) 0 0 (0)
Density: 71.588%
Routing Overflow: 0.00% H and 0.00% V
Reported timing to dir timingReports
Total CPU time: 1.51 sec
Total Real time: 1.0 sec
Total Memory Usage: 2156.027344 Mbytes
innovus 6> #-report cordic8bit.drc.rpt
*** Starting Verify DRC (MEM: 2156.0) ***
VERIFY DRC ..... Starting Verification
VERIFY DRC ..... Initializing
VERIFY DRC ..... Deleting Existing Violations
VERIFY DRC ..... Creating Sub-Areas
VERIFY DRC ..... Using new threading
VERIFY DRC ..... Sub-Area: {0.000 0.000 66.240 66.240} 1 of 4
VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {66.240 0.000 130.200 66.240} 2 of 4
VERIFY DRC ..... Sub-Area : 2 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {0.000 66.240 66.240 129.800} 3 of 4
VERIFY DRC ..... Sub-Area : 3 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {66.240 66.240 130.200 129.800} 4 of 4
VERIFY DRC ..... Sub-Area : 4 complete 0 Viols.
Verification Complete : 0 Viols.
*** End Verify DRC (CPU: 0:00:00.6 ELAPSED TIME: 0.00 MEM: 1.0M) ***
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
```

Figure 21 No DRC violations

## 11 Bugs known at submission date

The % errors reported in figure 21 cannot be minimized further because of the lack of resolution due to only 8 bits available for output.

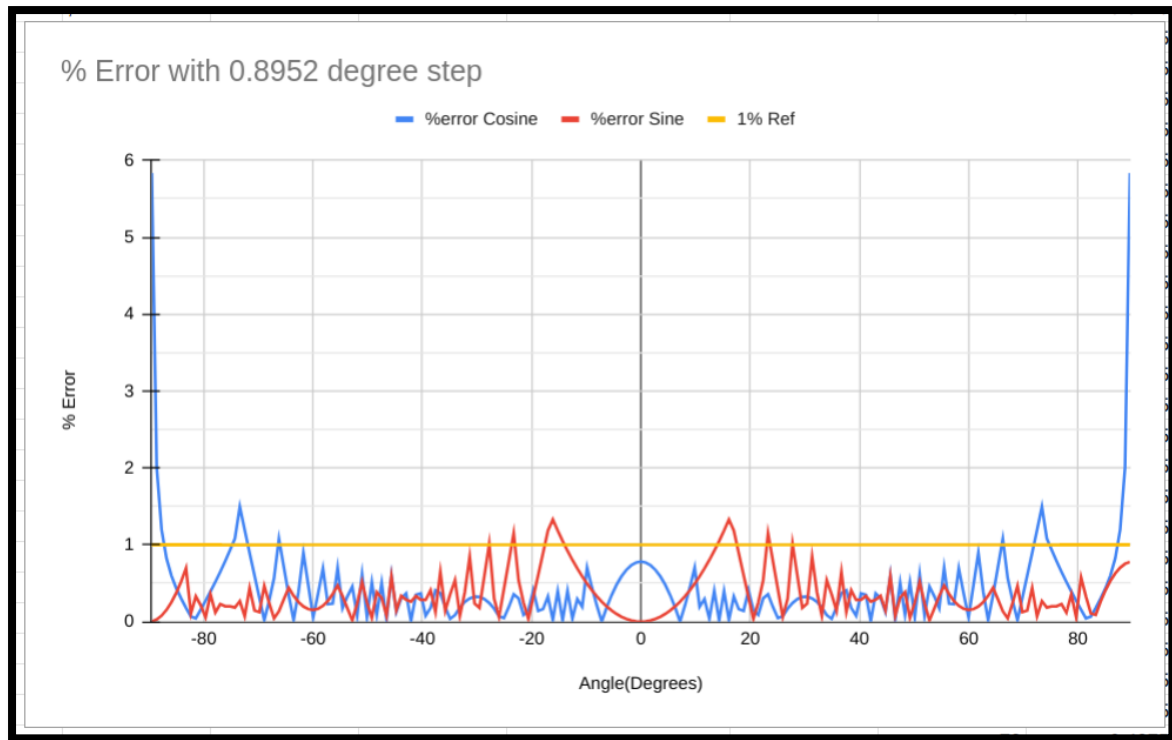


Figure 22 % error in sine and cosine of input angles Vs input angles

Table 5 Test Cases not meeting specifications

S.No	Radian	Angle (Deg)	Cosine	Sine	Correct Cosine	Correct Sine	%error Cosine	%error Sine
1	0.28125	16.11443799	0.960938	0.28125	0.960709243	0.2775567516	0.02381126091	1.33062818
2	1.5625	89.52465549	0.007812	0.992188	0.008296231624	0.9999655857	5.836765966	0.7777853348
3	1.546875	88.62940893	0.023438	0.992188	0.02391904544	0.9997138987	2.011139805	0.7528052491

Table 6 Explanation of not meeting test specifications

S.No			% Error	Original %Error
1	Original + 0_0000001	0.2890625	4.145367852	1.33062818
	Original - 0_0000001	0.2734375	1.484111491	
2	Original + 0_0000001	0.0156245	88.3324949	5.836765966
	Original - 0_0000001	-0.0000005	100.0060268	
3	Original + 0_0000001	0.0312505	30.6511168	2.011139805
	Original - 0_0000001	0.0156255	34.67339641	

