From C code:

Sum of errors for sin = **0.1105541363358498**

Sum of errors for cos = **0.1123068407177925**

From SystemVerilog Cordic Module:

Sum of errors for sin = **0.1116878507236145**

Sum of errors for cos = **0.1149921312188339**

The SystemVerilog Cordic module with quantization performed very similar to the floating-point software algorithm in C. The above values are the sums of the all the errors between the Cordic sin/cos values to the actual sin/cos values. The sum is taken by taking the absolute value of the differences as some of them were negative. The percent different is **1.025%** and **2.391%** for sin and cos respectively.

The actual Cordic outputs from the SystemVerilog code is in the textfile "test\_output.txt" in the "source" folder.

For the UVM simulations, the modules were the same but the testing using a scoreboard to compare DUT output to a set of comparable values was done a little differently. In my scoreboard, the Cordic sin and cos values were compared with the actual sin and cos values relative to a threshold of 0.001. The UVM would create a fatal error if any error exceeded that threshold.

The actual Cordic outputs from the UVM simulation is in the textfile "uvm\_test\_output.txt" in the "source" folder.

# ------------------------------------------------------------------

# Name Type Size Value

# ------------------------------------------------------------------

# uvm\_test\_top cordic\_uvm\_test - @359

# env cordic\_uvm\_env - @371

# agent cordic\_uvm\_agent - @379

# agent\_ap\_compare uvm\_analysis\_port - @413

# agent\_ap\_output uvm\_analysis\_port - @404

# drvr cordic\_uvm\_driver - @545

# rsp\_port uvm\_analysis\_port - @562

# seq\_item\_port uvm\_seq\_item\_pull\_port - @553

# mon\_cmp cordic\_uvm\_monitor\_compare - @579

# mon\_ap\_compare uvm\_analysis\_port - @589

# mon\_out cordic\_uvm\_monitor\_output - @571

# mon\_ap\_output uvm\_analysis\_port - @599

# seqr uvm\_sequencer - @422

# rsp\_export uvm\_analysis\_export - @430

# seq\_item\_export uvm\_seq\_item\_pull\_imp - @536

# arbitration\_queue array 0 -

# lock\_queue array 0 -

# num\_last\_reqs integral 32 'd1

# num\_last\_rsps integral 32 'd1

# sb cordic\_uvm\_scoreboard - @387

# compare\_fifo uvm\_tlm\_analysis\_fifo #(T) - @684

# analysis\_export uvm\_analysis\_imp - @728

# get\_ap uvm\_analysis\_port - @719

# get\_peek\_export uvm\_get\_peek\_imp - @701

# put\_ap uvm\_analysis\_port - @710

# put\_export uvm\_put\_imp - @692

# output\_fifo uvm\_tlm\_analysis\_fifo #(T) - @631

# analysis\_export uvm\_analysis\_imp - @675

# get\_ap uvm\_analysis\_port - @666

# get\_peek\_export uvm\_get\_peek\_imp - @648

# put\_ap uvm\_analysis\_port - @657

# put\_export uvm\_put\_imp - @639

# sb\_export\_compare uvm\_analysis\_export - @622

# sb\_export\_output uvm\_analysis\_export - @613

# ------------------------------------------------------------------

#

# UVM\_INFO /vol/mentor/questa\_sim-2019.3\_2/questasim/verilog\_src/uvm-1.2/src/base/uvm\_objection.svh(1270) @ 7635: reporter [TEST\_DONE] 'run' phase is ready to proceed to the 'extract' phase

# UVM\_INFO ../uvm/cordic\_uvm\_monitor.sv(63) @ 7635: uvm\_test\_top.env.agent.mon\_out [MON\_OUT\_FINAL] Closing file ../source/uvm\_test\_output.txt...

# UVM\_INFO /vol/mentor/questa\_sim-2019.3\_2/questasim/verilog\_src/uvm-1.2/src/base/uvm\_report\_server.svh(847) @ 7635: reporter [UVM/REPORT/SERVER]

# --- UVM Report Summary ---

#

# \*\* Report counts by severity

# UVM\_INFO : 7

# UVM\_WARNING : 0

# UVM\_ERROR : 0

# UVM\_FATAL : 0

# \*\* Report counts by id

# [MON\_OUT\_FINAL] 1

# [Questa UVM] 2

# [RNTST] 1

# [TEST\_DONE] 1

# [UVM/RELNOTES] 1

# [UVMTOP] 1

#

# \*\* Note: $finish : /vol/mentor/questa\_sim-2019.3\_2/questasim/verilog\_src/uvm-1.2/src/base/uvm\_root.svh(517)

# Time: 7635 ns Iteration: 70 Instance: /cordic\_uvm\_tb

# End time: 16:56:26 on Feb 24,2024, Elapsed time: 0:00:01

# Errors: 0, Warnings: 0

Simulation Cycle Count = **764 cycles**(UVM simulation)

Simulation Total Time = **7635 ns**(UVM simulation)

For 100MHz:

Sampling Rate = **100,000,000 angles/second** (1 angle is "sampled" / 10 ns)

Throughput = **94,302,554 angles/second**(720 angles / 7635 ns or 0.000007635 s)

Max Frequency = **286.1 MHz**

Total combinational functions 121 of 6272 ( 1%)

Logic element usage by number of inputs

4 input functions 52

3 input functions 39

[=2 input functions 30

Logic elements by mode

normal mode 103

arithmetic mode 18

Total registers 39 of 6272 ( 0%)

I/O pins 72 of 180 (40%), total I/O based on largest package of this part.

Number of I/O registers

Input DDRs :0

Output DDRs :0

DSP Blocks: 0 (0 nine-bit DSP elements).

DSP Utilization: 0.00% of available 15 blocks (30 nine-bit).

ShiftTap: 0 (0 registers)

Ena: 0

Sload: 0

Sclr: 0

Total ESB: 2048 bits

Black-boxes: 1

cordic\_16s\_16s\_9949s\_51472s\_25736s\_102944s\_Z1 : 1

A screenshot of a computer

Description automatically generated

Above is the technology schematic of my cordic\_top module that has an input FIFO to hold the 32-bit quantized angles in radians that feed into the cordic\_top module. There are also 2 output FIFOs to hold the 16-bit quantized sin and cos values from the Cordic algorithm. The Cordic module is represented as a black box so no schematics from inside can be observed.

After an angle is inputted, the Cordic module checks if it's in between -Pi and Pi and adjusts the angle if not to ensure that it is within that range. Then it adjusts the starting x vector direction and the angle again to make sure the vector is inside the 1st or 3rd quadrants of the polar coordinate graph.

The starting x and y parts of the vector and the starting angle is then fed into the first cordic\_stage module which performs the Cordic algorithmn. Its outputs are then passed to the next cordic\_stage module until the 16th one where the Cordic values of sin and cos are pushed into the output FIFOs.

All the quantization is done on the test bench level so the outputs of the Cordic module is a 16-bit quantized sin or cos value that has to be de-quantized before being written to a file or printed. The input angle is quantized but remains in 32-bit since there is some 32-bit arithmetic necessary to ensure the angle and vector are within the 1st or 3rd quadrants.