Efficient fault diagnosis for CORDIC Algorithm by using Hamming Codes

Prachi Jain¹, Dr. Ananiah Durai²*
School of Electronics Engineering, VIT, Chennai¹,²
Email:²anianahdurai.s@vit.ac.in²

Abstract. The attractiveness of CORDIC Algorithm lies in the fact that by simple shift-add operations, it can perform several computing tasks such as the calculation of logarithmic, hyperbolic and trigonometric functions, real and complex multiplications, square-root, Eigen value estimation and many others. As a consequence, CORDIC has been used diversely for applications in diverse areas such as communication systems, 3-D graphics, signal and image processing apart from general scientific and technical computation. The digital computation is very much prone to errors; hence the error detection and correction became important for this algorithm. In this work it is achieved by integrating the Hamming codes to the CORDIC algorithm and hence achieving the error-free CORDIC algorithm. The proposed design are benchmarked through Xilinx FPGA and hence the reliability constraints be achieved.

1. Introduction
In the recent year, the need for information transfer over a network of CORDIC algorithm with optimum efficiency has been of increasing demand due to its simple processing and hardware. CORDIC (Coordinate Rotation Digital Computer) is also known as Volder's algorithm. It is an efficient and simple algorithm which uses simple shift-add operations for several computing tasks such as the calculation of trigonometric, logarithmic and hyperbolic functions, complex and real multiplications, division, solution of linear systems, square-root calculation, Eigen value estimation, singular value decomposition, QR factorization and many others, typically converging with one digit (or bit) per iteration henceforth belong to the class of shift-and-add algorithms. CORDIC is hence also an example of digit-by-digit algorithms.

Therefore consequently, CORDIC has been used for applications in diverse areas such as signal and image processing, communication systems, 3D graphics and robotics and many more apart from general scientific and technical computation. The digital computation is very much prone to errors; hence the error detection and correction became important for this algorithm. Error detection and correction is done with the help of Hamming Codes. Hamming codes is a Forward Error-correcting Code (FEC) that uses redundant bits to correct a single bit error. It consists of the set of error-correction codes that can be used to detect the errors that can occur when the data is moved or stored from the sender to the receiver (shown in Figure 1).
4 bit codes, 3 redundant bits are needed so total = 7 bits. For 8 bit codes, 4 redundant bits are needed so total = 12 bits. They are placed in bit positions 1, 2, 4 and 8 (powers of 2) and the 8 bit character occupies bit positions 3, 5, 6, 7, 9, 10, 11 and 12.

Each redundant bit is the Vertical Redundancy Check (normally even parity) for a combination of data bit. Hamming code can be applied to data of any length and use the relationship between data and redundancy bits. For introducing noise in Cordic algorithm for our work, so to make it a real-time output consisting of noise, LFSR is being used. An LFSR is a shift register that, which when clocked, advances the signal through the register from one bit to the next most-significant bit. Some of the outputs are combined in exclusive-OR (XOR) configuration to form a feedback mechanism. LFSR is formed by performing an exclusive-OR on the outputs of two or more flip-flops together and feeding those outputs back into the input of one of the flip-flop.

An example of LFSR is shown in Figure 2 below:

2. Existing Work
Survey study focused on the error detection schemes for fixed-angle rotation CORDIC designs, where stuck-at fault model are considered in which the nodes in the architectures are stuck-at zero or one, regardless of their correct values [1].

In this paper the survey study focused on a unique computing technique, which is especially suitable for solving the conversions from rectangular to polar coordinates and trigonometric relationships in plane coordinate rotation. By using the conditional additions or subtractions, the CORDIC arithmetic unit is been controlled [2].

The focus was to propose error detection schemes for CORDIC architectures used vastly in various applications such as complex number multiplication, and singular value decomposition for signal and image processing [3].
The objective was to present a brief overview of the key developments in the CORDIC algorithms and architectures along with their potential and upcoming applications [4]. The objective of this paper was to firstly classify the CORDIC algorithm which is based on the number system and discuss its importance in the implementation of CORDIC algorithm, and then presents the comprehensive and systematic taxonomy of rotational CORDIC algorithms, which are subsequently explained in depth. The comparison of various algorithms is been presented at the end of work, which provides a first-order information to designers looking for either further improvement of performance or selection of rotational CORDIC algorithm for a specific application [5].

3. Design and Methodologies

3.1 Error Detection:
CORDIC Algorithm is a digital algorithm and hence is prone to errors. So, in this step to acquire the real-time output of the Cordic Algorithm, LFSR (linear-feedback shift register) is used to introduce some noise in the network since LFSR provides pseudo-random bits. On introducing LFSR to the Cordic Algorithm erroneous outputs are generated, and hence is detected by the Error detection block. The error detection block shown in Figure 3, mostly uses linear function of single bit exclusive-or (XOR) logic.

3.2 Error Correction:
The output obtained from the Error correction block is then being fed to the input of Hamming block, which is shown in Figure 4. Hamming block is used for bit-to-bit error correction and hence retracing the expected results of CORDIC Algorithm.
After the deduction of results through simulation waveforms, the percentage deviation of the bits is being calculated.

4. Implementation Environment
The implementation is carried out in Xilinx FPGA platform for synthesis and analysis of HDL design. The Verilog HDL code is written for selected algorithms and synthesized in the Xilinx platform. The vertex 6 low power family is chosen in the design properties.

5. Results and Discussion
The results are obtained after the synthesis of the above methodology as shown in Figure 5. The obtained results are tabulated in Table 1.

![Figure 5. Cordic Algorithm output](image)

The simulation outputs are as expected output for a Cordic algorithm; while analysing, the outputs are found to be without any error.

| Cordic input | Cordic (expected) output |
|--------------|--------------------------|
| Xin= 0100101111100101 | Xout= 0010100010100001 |
| Yin= 0000000000000000 | Yout= 0000111100100110 |
| Angle= 11111111010010011111010010011111 | |
| Xin= 0100101111100101 | Xout= 0010110001010001 |
| Yin= 0000000000000000 | Yout= 0010110011011111 |
| Angle= 00100000000000000000000000000000 | |
| Xin= 0100101111100101 | Xout= 0111000111110110 |
| Yin= 0000000000000000 | Yout= 0001101111111001 |
| Angle= 00010010111001000000010100011101 | |
| Xin= 0100101111100101 | Xout= 0010011101100101 |
| Yin= 0000000000000000 | Yout= 0000011111100010 |
Simulation output in Figure 6 depicts the output of Cordic algorithm in the presence of noise. To insert the system noise LFSR is been used. Hence the erroneous deviated output is been observed. The output of the detection block is tabulated in Table 2.

**Table 2**: The output of Error Detection Block

| Inputs of Error Detection block | Outputs of Error Detection block |
|---------------------------------|----------------------------------|
| Xout= 00101000101000011          | Sout= 00111100101000100          |
| Yout= 00001111001001111          | Pout= 00001111001001010          |
| Out= 0000000000000000111         |                                  |
| Xout= 00101100001010111          | Sout= 00111100101000100          |
| Yout= 00101000010111111          | Pout= 00101100001011100          |
| Out= 000000000000000001111       |                                  |
| Xout= 00100011011100110          | Sout= 00111100101000100          |
| Yout= 00000111111111000100       | Pout= 00001111001111011          |
| Out= 000000000000000001111       |                                  |
| Xout= 00111100010111010          | Sout= 00111100101000100          |
| Yout= 00101100111100110          | Pout= 00001111001111011          |
| Out= 0000011111000010010         |                                  |
| Xout= 00111100101110101          | Sout= 00111100101000100          |
| Yout= 00101101110010110          | Pout= 00001111001111011          |
| Out= 0000011111000011111         |                                  |

### 6. Calculations of bit deviation

From table 2 percentage bit deviation can be calculated from (1) provided below.

\[
\text{\% Deviation} = \frac{\text{Deviation between input and output bits}}{\text{Total number of bits}} \times 100
\]  

(1)

The calculations of deviations for different outputs of error detection block listed in column 1 and 2 of Table 1 are given below:
For the values in row 1 of Table 1, the deviations are:
- % Deviation in Xout = 29.41%
- % Deviation in Yout = 17.64%

For the values in row 2 of Table 1, the deviations are:
- % Deviation in Xout = 11.76%
- % Deviation in Yout = 29.41%

For the values in row 3 of Table 1, the deviations are:
- % Deviation in Xout = 52.94%
- % Deviation of Yout = 29.14%

For the values in row 4 of Table 1, the deviations are:
- % Deviation in Xout = 29.41%
- % Deviation in Yout = 52.94%

For the values in row 5 of Table 1, the deviations are:
- % Deviation in Xout = 47.65%
- % Deviation in Yout = 23.52%

The Bit deviation is calculated between expected output and the obtained output of Cordic algorithm. From the above calculation the deviation of bits can be observed. Bit deviation cannot be entertained when the Cordic algorithm is used in various real time applications. Hence Hamming codes is being used for error detection and correction.

Figure 7. Error Correction
The above Figure 7 denotes that on inserting the Hamming block to the erroneous output of Cordic algorithm, the error has been detected and corrected and hence the expected correct output can be obtained.

Table 3: The output of Hamming block

| Hamming block inputs | Hamming block outputs |
|----------------------|-----------------------|
| Sout= 01111100101000011 | Xcrr= 00101100001011101 |
| Pout= 00000110010101101 | Ycrr= 00001111001100011 |
| Out1= 00000000000000111 |

As compared from Table 1, the outputs obtained in Table 3 after insertion of Error correction Hamming block are similar, that is, Xout is coming similar to Xcrr and similarly Yout is coming similar to Ycrr without any bit deviation.

7. Conclusion
The research basically aimed at finding the efficient and reliable constraint for CORDIC Algorithm. The CORDIC algorithm is based on digital computation, hence is very much prone to errors, therefore the error detection and correction became important for this algorithm. In this work it is achieved by integrating the Hamming codes to the CORDIC algorithm and hence achieving the error-free and hence reliable CORDIC algorithm results.

8. References
[1] R Ramadoss, M Mozaffari Kermani and R Azarderakhsh 2017 Reliable Hardware Architectures of the CORDIC Algorithm With a Fixed Angle of Rotations IEEE Transactions on Circuits and Systems II: Express Briefs pp 972-76.
[2] J E Volder 1959 The CORDIC Trigonometric Computing Technique IRE Transactions on Electronic Computers pp. 330-34.
[3] M M Kermani, R Ramadoss and R Azarderakhsh, 2016 Efficient error detection architectures for CORDIC through recomputing with encoded operands IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC pp. 2154-157.
[4] P K Meher, J Valls, T Juang, K Sridharan and K Maharatna, 2009 50 Years of CORDIC: Algorithms, Architectures, and Applications IEEE Transactions on Circuits and Systems I: Regular Papers pp. 1893-1907.
[5] B Lakshmi and A S Dhar 2010 CORDIC architectures: a survey VLSI Design 74891.