International Journal of Microcircuits and Electronic.
ISSN 0974-2204 Volume 4, Number 1 (2013), pp. 1-9

http://www.irphouse.com

# FPGA Design, Implementation and Analysis of Trigonometric Generators using Radix 4 CORDIC Algorithm

**J M Rudagi* and Dr shaila subbaraman* ***

*js_itti@yahoo.co.in*
**Dept of Electronics and Communication, KLECET, Belgaum, Karnataka, India.*
*** Dept of Electronics and communication ,*
*Dange college of engineering, Maharashtra.*

**Abstract**

Today, many of the computations in signal processing and wireless communication applications are linked with complex analysis of several functions. These complex functions are combination of sine and cosine terms that generally spread in the channel. Most of these functions can be split into elementary functions. Today user's desire every gadget must be smaller in size and simple to operate ("Keep it small and simple"). So the researchers muddle through speed area tradeoff. The CORDIC algorithm is one such kind of algorithm. In this paper hardware efficient trigonometric generator is designed and implemented on Spartan 3 FPGA. The results obtained are compared with that obtained using MATLAB.

**Keywords:** CORDIC, FPGA, VLSI, Sine, Cosine, Radix 4, pipelined architecture, latency.

## 1. Introduction

The key concept of CORDIC architecture is based on the simple and the ancient concept of the two dimensional geometry. CORDIC algorithm is originally proposed by Volder [1] in1959 and later generalized by Walther [2] in 1971 for computation of logarithms, exponentials and square root functions along with trigonometric functions like sine, cosine and tangent. It is an iterative algorithm for the calculation of the rotation of two dimensional vectors in linear, circular and hyperbolic coordinate systems. The current applications of CORDIC algorithm are in the field of DSP, image processing, filtering, matrix algebra etc. The algorithm belongs to the class of

linear convergence algorithms and can likewise be implemented using only shift and add operations making it suitable for VLSI implementations.

The number of iterations of conventional radix 2 CORDIC architectures limits it to be used in a high speed application. So the development of high radix CORDIC algorithms is essential for reducing the number of iterations

Reduction of latency with reasonable increment in hardware complexity. The scale factor overhead causes its improvement to be limited. The minimization of the computational overhead of scale factor compensation been attempted by different researchers. [7-11].

In this paper a folded radix 4 CORDIC architecture is implemented. The total number of iterations is reduced by half (n/2). The scale factor compensation is carried out in parallel to the rotation. The architectures are implemented on available FPGA. The results are compared with MATLAB output.

The work is structured as follows. Section 2 defines the radix 4 CORDIC algorithm along with its scale factor compensation. Section 3 describes folded architecture, based on the radix 4 CORDIC algorithm. Section 4, covers implementation details. Section5.Deals with comparative study. Concluding remarks are presented in section 6.

## 2. Radix 4 CORDIC algorithm:

The radix 4 CORDIC equations are [4],

$$X_{i+1}=X_i-\sigma_i 4^{-i}Y_i$$
$$Y_{i+1}=Y_i+\sigma_i 4^{-i}X_i \quad (1)$$
$$Z_{i+1}=Z_i-\alpha_i[\sigma_i]$$

Where $\sigma_i \varepsilon \{-2,-1, 0, 1, 2\}$, $\alpha_i [\sigma_i] \varepsilon \tan^{-1} (\sigma_i r^{-i})$, where r is the radix of the CORDIC. $X_{i+1}$ , $Y_{i+}1$ are the coordinates the vector resulting from applying i+1 micro rotation and $Z_{i+1}$ is the angle to be rotated. The coordinates are scaled by ,

$$K^{-1}=\prod_{i\geq 0}^{n/2-1} (1+\sigma_i^2 4^{-2i})^{-1/2} \quad (2)$$

The scale factor depends on $\sigma_i$. For radix 4 algorithm this value ranges from K=1.0 to K=2.52[9].This scale factor must be evaluated for each rotation angle and compensated.

Let us define a variable $w_i$ as,

$$w_i=4^i Z^i \quad (3)$$
$$A_i[\sigma_i]=4^i \tan\text{-}1(\sigma_i * 4^{-i})$$

Iteration Z can be expressed as,

$$W_{i+1}=4(w_i- A_i[\sigma_i]) \quad (4)$$

The Eq. (4) is required to prove that the variable Z is bounded in each of the iterations.The selection interval for different values of iteration of eq. (1) can be obtained [4],

For i=0,

$$\sigma_0 \begin{cases} +2 \text{ if } 5/8 \leq \hat{W}_0 \\ +1 \text{ if } 3/8 \leq \hat{W}_0 < 5/8 \\ 0 \text{ if } -1/2 \leq \hat{W}_0 < 3/8 \\ -1 \text{ if } -7/8 \leq \hat{W}_0 < -1/2 \\ -2 \text{ if } \hat{W}_0 < -7/8 \end{cases} \quad (5)$$

For i>0,

$$\sigma_i \begin{cases} +2 \text{ if } \hat{W}_i \geq 3/2 \\ +1 \text{ if } 1/2 \leq \hat{W}_i < 3/2 \\ 0 \text{ if } -1/2 \leq \hat{W}_i < 1/2 \\ -1 \text{ if } -3/2 \leq \hat{W}_i < -1/2 \\ -2 \text{ if } \hat{W}_i < -3/2 \end{cases} \quad (6)$$

Here $\hat{W}_i$ is the estimate of $w_i$ with three most significant bits. The selection function is true for both carry save redundant arithmetic and non redundant arithmetic [14].

**2.1 Scale factor**

The final coordinates obtained from the application of the micro rotations (1) do not match the result of rotation. The final coordinate are scaled by a scale factor (2). This factor is not constant as it depends on the sequence of $\sigma_i$'s and has to be calculated for each rotation angle. To provide compensation we have to calculate scale factor for each angle and perform direct multiplication to obtain the X and Y coordinates of the rotated vector.

The scale factor can be calculated for n/4+1 micro rotation as for rest of the micro rotations it can be taken as one. The scale factors can be stored in a table. The size of the table is $(3^{n/4+1}*n)$. The Taylor series expansion of (2),

$$K^{-1}=1-1/2*\sigma_i^2*4^{-2i}+3/8*\sigma_i^4*4^{-4i}+\cdots\cdots \quad (7)$$

The $K^{-1}$ can be approximated by the first two terms for $i \geq [n/8+1]$ or three terms if $i \geq [n/12+1]$. Hence we can store only scale factor generated in the first [n/12+1] micro rotations, since the scale factor generated by micro rotations with $i \geq [n/12+1]$ can be calculated by means of addition and shift operations. The size of the table can be reduced to $(3^{n/12+1} * n)$.

## 3. Folded Architecture: [4] The architecture is designed for 32 precision.

### 3.1Unscaled CORDIC architecture

This architecture consists of a preprocessing unit for carrying out π/2 rotations and three processing paths for X, Y, and W coordinates. It requires sixteen iterations to complete its operation. Every clock cycle produces intermediate values of X and Y.

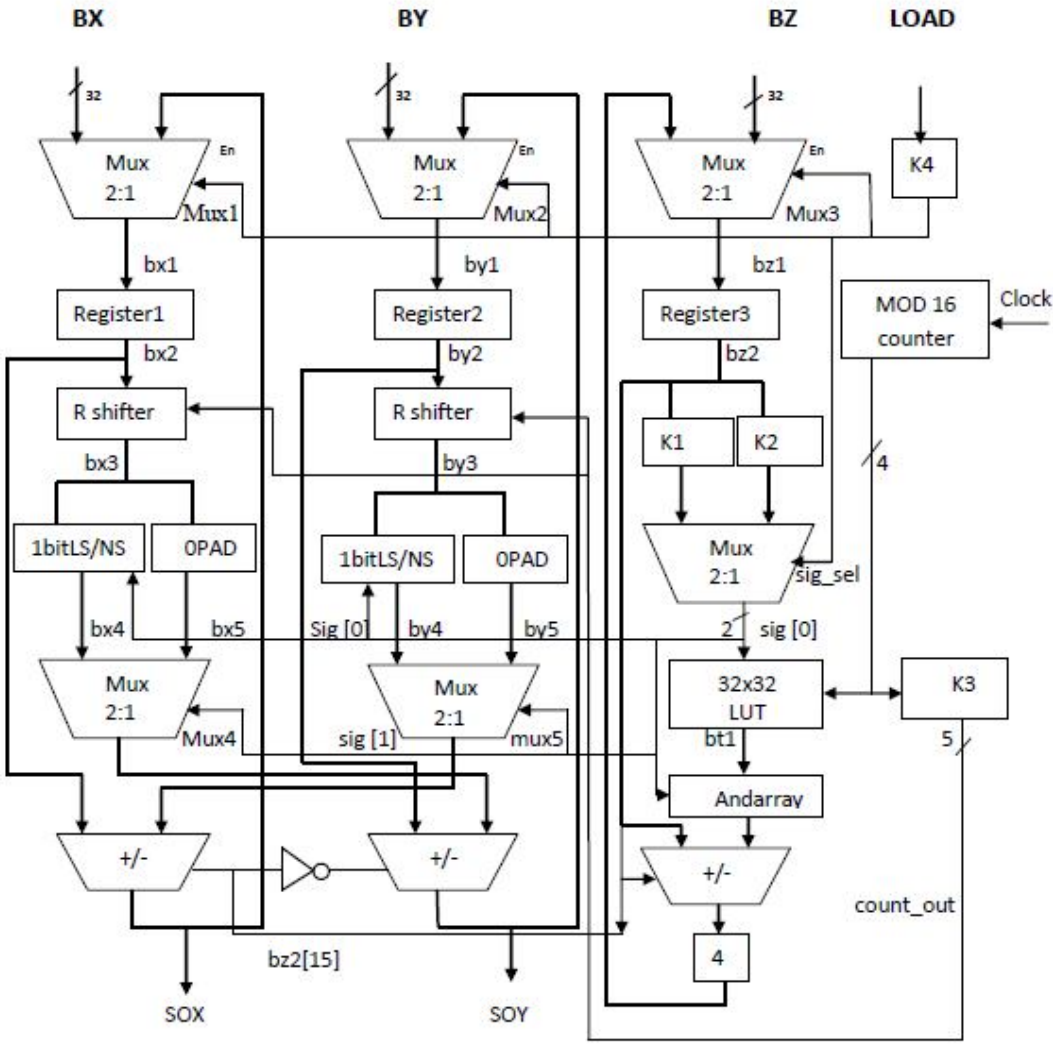


Fig 1: "Architecture of un scaled folded 32 bit CORDIC processor"

### 3.2 $K^{-1}$ Calculation

The scale factor is calculated in parallel with X, Y and W coordinates. It requires nine cycles (i.e. n/4+1).

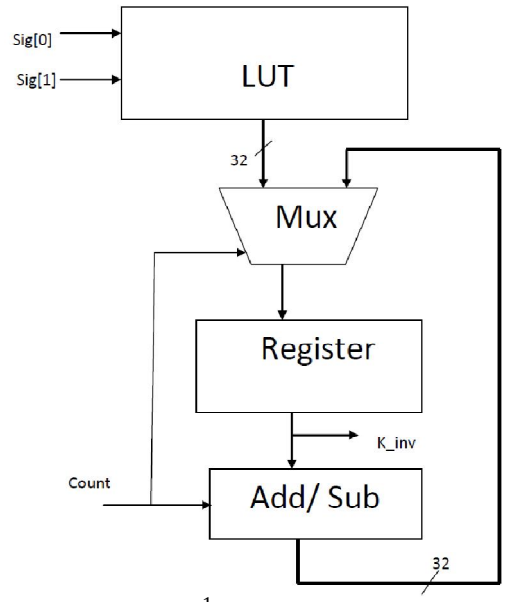


Fig 2: "$K^{-1}$ Computation"

### 3.3 Scaled 32bit CORDIC architecture

Two 32bit multipliers are used to provide compensated values of X and Y.

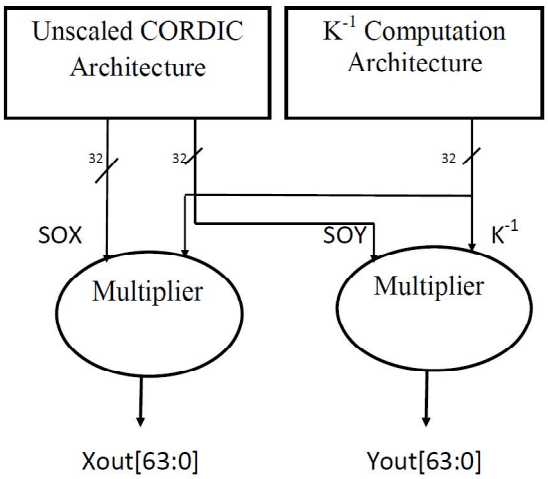


Fig 3: "Architecture of scaled folded 32 bit CORDIC processor"

## 4. Implementation

The CORDIC algorithm is verified using MATLAB tool (version7.10.0.499, R2010a ) . The architecture is implemented on the available XC3S400-4PQ208 of the XILINX FPGA (ISE 10.1i). The proposed architecture has been modeled in VERILOG. The combinational path delay of folded 32bit architecture is 33.306ns. The latency of the folded 32bit architecture is n/2(i.e. 16) clock cycles. The

throughput rate is one valid result per sixteen clock cycles. The folded 32bit architecture operates at 55.906MHz of the clock rate. The resource utilization of FPGA utilization for the architecture is given in Table 1.

Table 1. Resource utilization by the folded 32bit CORDIC architecture on the target device XC3S400-4PQ208.

| Device Utilization Summary | | | | |
|---|---|---|---|---|
| Logic Utilization | Used | Available | Utilization | Note(s) |
| Total Number Slice Registers | 133 | 7,168 | 1% | |
| Number used as Flip Flops | 102 | | | |
| Number used as Latches | 31 | | | |
| Number of 4 input LUTs | 1,012 | 7,168 | 14% | |
| Logic Distribution | | | | |
| Number of occupied Slices | 542 | 3,584 | 15% | |
| Number of Slices containing only related logic | 542 | 542 | 100% | |
| Number of Slices containing unrelated logic | 0 | 542 | 0% | |
| Total Number of 4 input LUTs | 1,035 | 7,168 | 14% | |
| Number used as logic | 1,012 | | | |
| Number used as a route-thru | 23 | | | |
| Number of bonded IOBs | 70 | 141 | 49% | |
| Number of MULT18X18s | 8 | 16 | 50% | |
| Number of BUFGMUXs | 2 | 8 | 25% | |

The CORDIC employed uses circular co-ordinate system and is operated in rotating mode. Hence, only phase is given as input and x, y values are given internally in the program or hardware. The code written in verilog is simulated using modelsim XE III 6.3c.

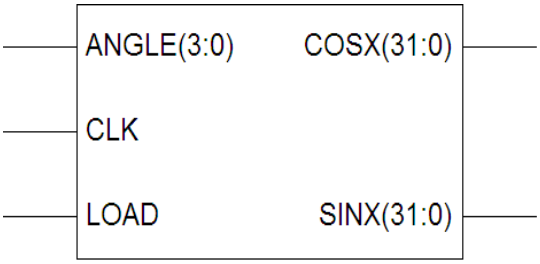


Fig.4 “Top Level RTL Schematic”

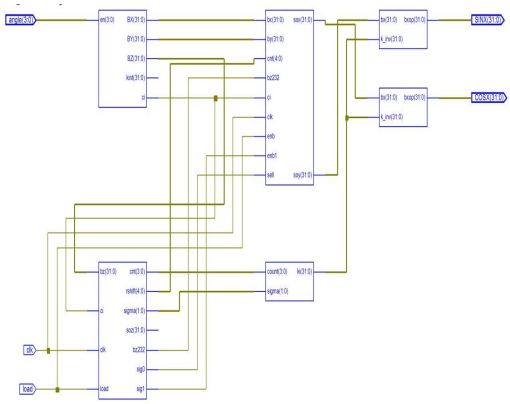

Fig.5. “RTL Schematic of scaled folded 32 bit CORDIC architecture”

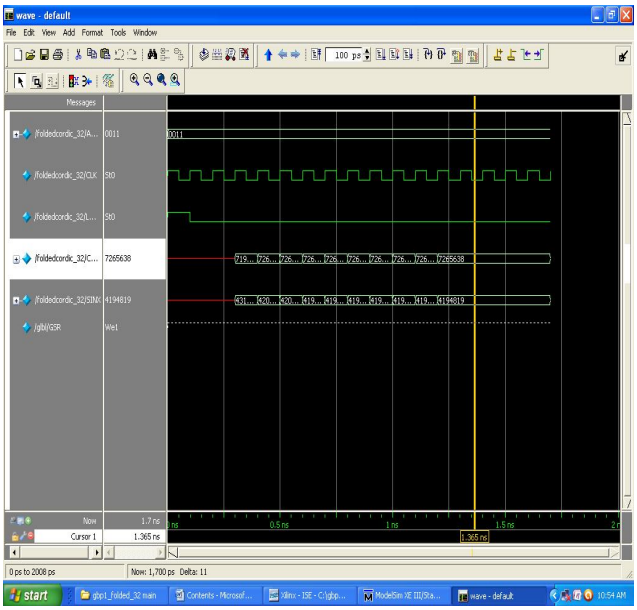


Fig 6. Simulation result

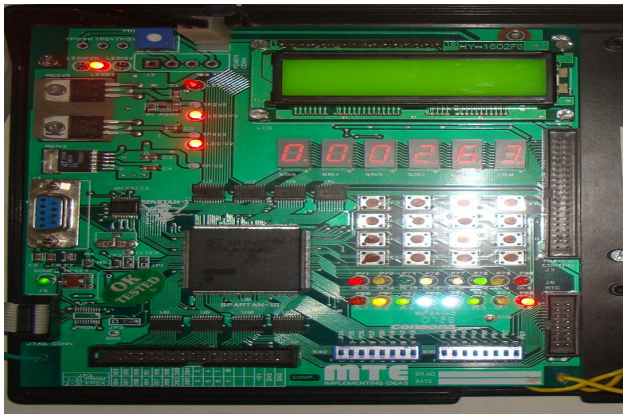

Fig 7. FPGA Implementation.

## 6 Comparative study

The simulation results obtained are compared with that of MATLAB outputs. It is found that the percentage error is within

$\pm$ 0.21%.

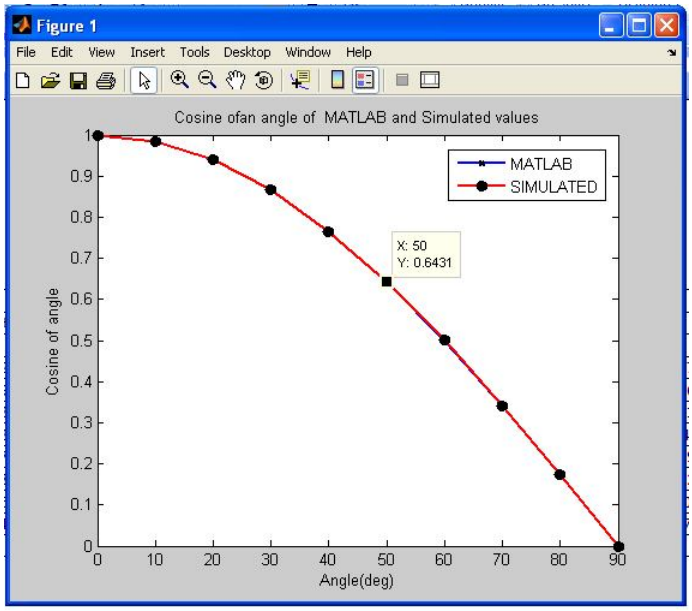


Fig.8. Comparison of MATLAB results with simulated results

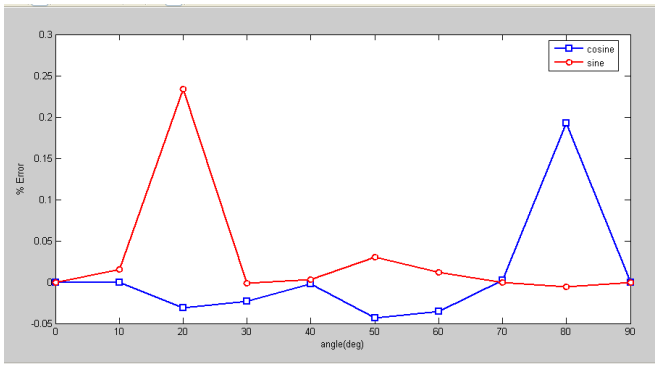


Fig.9. Angle vs. % Error

## 6. Conclusion

The folded CODIC architecture has been implemented on the available XILINX FPGA. The folded architecture operates at a frequency of 55.906MHz. Latency of the architecture is n/2 (i.e.16) clock cycles. The sine and cosine values generated by the architecture are comparatively accurate. The Trigonometric generators with optimized speed area can be used for real time applications.

## References

[1] J.E. Volder, “The cordic trigonometric computing technique”, IRE Trans Electronic Computers 8 (1959) 330–334.

[2] J.S. Walther, “A unified algorithm for elementary functions”, in: Proceedings of Spring. Joint Computer Conference, 1971, pp. 379–385.

[3] E. Antelo et al., J. Villaba,J.D brugura,E L Zapata “High performance rotation architectures based on radix-4 CORDIC algorithm”, IEEE Transactions Computers 46 (8) (1997) 855–870.

[4] Kaushik Bhattacharyya , Rakesh Biswas, Anindya Sundar Dhar, Swapna Banerjee, “Architectural design and FPGA implementation of radix-4 CORDIC processor”, Microprocessors and Microsystems 34 (2010) 96–101

[5] K.Hwang, “Computer Arithmetic Principles, Architecture and Design”, Wiley, NewYork, 1979

[6] Takafumi et al., “High Radix CORDIC algorithm for VLSI signal processing”, in IEEE Workshop on Signal Processing Systems (SIPS), November 1997, pp. 183–192

[7] M.G. Buddika Sumanasena, “A scale factor correction scheme for the CORDIC algorithm”, IEEE Transactions on Computers 57 (8) (2008) 1148–1152.

[8] Villaba .J Lang et al., “CORDIC architectures for the parallel compensation of scale factor”, in: Poc , International conference on Application Specific Array Processors , ASAP’95, July 95 , pp 258-269.

[9] H.X.Lin, H.J.Sips, “Online CORDIC algorithms”, IEEE Transactions on Computers 39(8) (1990) 258-269.