REVIEW OF CORDIC ARCHITECTURES

T.SUBHA SRI [#1], K.SESHU KUMAR [#2], R. MUTTAIAH[#3]
School of Computing, M.Tech VLSI design, SASTRA University
Thanjavur, Tamil Nadu, 613401, India
luckysanju8@gmail.com [#1]
seshukumar6039@gmail.com [#2]

***Abstract*— Coordinate Rotation Digital Computer (CORDIC) first introduced by volder and Jack E. The consequence is lies on reality i.e., Small operations of shift-add and it also perform numerous computation operations. CORDIC used in various applications like image and signal processing, robotics systems and also communication systems and technical totalling. In this paper we presenting a brief outline and also growth in the CORDIC algorithm and architectures.**

Keyword- CORDIC Algorithm; CORDIC Architectures

I. INTRODUCTION

In 1956 Jack Volder introduced CORDIC algorithm, it was most effective, robust and low difficulty method to calculate many functions [1]. The CORDIC used wide-ranging applications; those are numerical coprocessors, pocket calculators, and radar signal processing. Later on CORDIC became replacement for the analog navigation computers system used in aircraft. CORDIC airborne navigational computer constructed to outperform the conservative modern of PCs by a factor of 7, and also binary to BCD number representation. In 1971 Walther [2,3] introduced procedure to calculate rotations in the manner of hyperbolic and circular methods and also linear coordinate systems. Afterwards, CORDIC is best choice for scientific calculator and also used in HP- 9100 and HP-35 desktop calculators, and HP-2152A co-processor [4, 3]. The CORDIC arithmetic processor will perform all the rotation. Later on algorithm used in linear transformations [6], for DSP applications introduced CORDIC processor with single chip [5], digital filters [7], and for signal processing algorithms introduced matrix based methods [8]. Now a days used VLSI technology and EDA tools also in the neural networks [9], and software defined radio [10], field of biomedical signal processing.Though CORDIC is not fastest method to perform all these operations, due to some reasons like low cost and efficient implementation it is most attractive. For real time DSP computations they did several modifications in CORDIC algorithm from past 20 years to deliver low cost and high performance hardware solutions. These have mainly focused on latency reduction and thus increase the CORDIC algorithm throughput.

Hekstra proposed latest arithmetic operation called as orthonormal rotations or fast rotations by use of fixed angles. This will performs the least amount of shift add operations these are required for rotations. Vander Kolkand Lee said that appropriate fast rotation in the effective and fast selection and it shows the advantage to be extended when applied to (EVD) Enhanced Versatile Disc. Lang and Bruguera [11] introduced a higher radix method instead of radix-2 method, for this operation requires less iteration. The higher radix method mainly suggests that the constant scale factor. They did on-line calculation and subsequent compensation logarithm of the scale factor. Lau, and Delosme [12] introduced 4-D householder CORDIC transform instead of using multi-dimensional transform. This is done by using 2-D CORDIC operations in sequence manner; it is used in singular value deposition (SVD). Swartzlander and Choi [13] introduced sign prediction and addition to overawe the critical path. This done by calculating both outcomes of the sign, and finally it will select the suitable one at the end of final iteration.

II. CORDIC ALGORITHM

CORDIC offers vector rotations of arbitrary angles, iterative process performed by using only shift and add operations. The Volder algorithm [1], is resulting from the general rotation transform [14]

$$x \sin ø + y \cos ø = y' \quad (1)$$

$$x \cos ø - y \sin ø = x' \quad (2)$$

This Cartesian plane rotates by *ø* angle.

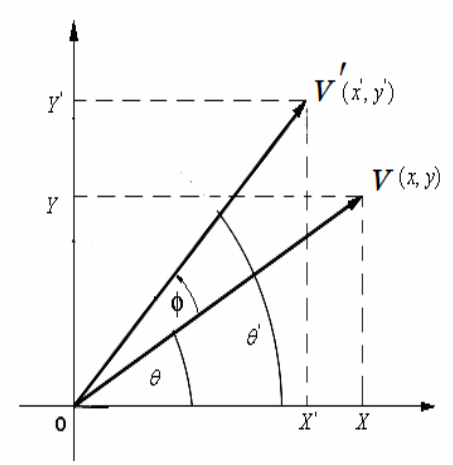

Fig 1: Vector rotation by ø angle.

The above equations can readjusted as:

$$[\, y + x \tan ø \,] \cos ø = y' \quad (3)$$

$$[\, x - y \tan ø \,] \cos ø = x' \quad (4)$$

These rotation of angles constrained so $\tan ø = \pm 2^{-i}$. This will reduces tangent multiplication by simple shift operation. Another rotation of arbitrary angles can easily accessible by the execution of elementary rotations in simple way. CORDIC, however, it will implement the rotations by a order of micro-rotations by elementary angles (α_i). These are obtained by decomposing ø into elementary rotations in sequence manner:

$$ø = \Sigma\, \alpha_i \quad (5)$$

By using these we can form simple iterative rotations

$$[y_i + x_i \tan \alpha_i] \cos \alpha_i = y_{i+1} \quad (6)$$

$$[\, x_i - y_i \tan \alpha_i \,] \cos \alpha_i = x_{i+1} \quad (7)$$

From trigonometric identities we have:

$$\cos \alpha_i = 1/ (1+\tan 2\, \alpha_i)^{1/2} \quad (8)$$

Substituting equation (8) in equation (6) and (7) we have:

$$[\, y_i + x_i \tan \alpha_i \,] / (\, 1 + \tan 2\, \alpha_i \,)^{1/2} = y_{i+1} \quad (9)$$

$$[\, x_i - y_i \tan \alpha_i \,] / (1 + \tan 2\alpha_i \,)^{1/2} = x_{i+1} \quad (10)$$

To assure that the tangent multiplication reduced to small shifting operation, the rotation angles obtained from following relation:

$$\tan \alpha_i = \pm 2^{-i} \quad (11)$$

Here *i* is total number of iteration. And $\tan \alpha_i$ replaced in above equation Substituting equation (12) in (9) and (10), we have:

$$[\, y_i + x_i \,.d_i.\, 2^{-i}].K_i = y_{i+1} \quad (12)$$

$$[\, x_i - y_i \,.d_i.\, 2^{-i}].K_i = x_{i+1} \quad (13)$$

Here,

$1/ (1+2^{-2i})^{1/2} = Ki$, denoted as constant scale.

$di = \pm 1$, it's denoted as decision function.

Shift-add algorithm eliminates the constant scale value from yields of obtained equations of vector rotation. Product term K_i's is applied in the system. The product value reaches to 0.6073 by using infinite number of iterations. A_n denoted as rotation algorithm gain,

$$An = \Pi\, [1+2^{-2i}]^{1/2} \quad (14)$$

For infinite iterations, the gain is approximately equal to 1.647. However gain can be calculated by the help of total number of iterations, *and* also it depends on composite rotation angle is distinct by sequence of elementary rotations in the directions manner. These sequences denoted as decision vector. And all these vectors used for angular measurement system, based on the values of binary arctangents. Conversions done by using a look-up table between angular system and any other systems. An improved conversion technique uses additional subtract or adder unit, for each iteration it mount up the elementary rotation angles. These angels are expressed by

convenient angular unit. A small lookup table supplies the angular values, and also we can use hardwired, depending on suitable implementation.

Accumulator angle can add the 3rd value of difference equation to CORDIC:

$$z_i - \alpha_i = Z_{i+1}$$
$$z_i - d_i \tan^{-1}(2^{-i}) = Z_{i+1}$$

Thus, only one CORDIC micro-rotation equations can be written as:

$$x_i - y_i . 2^{-I} . d_i = x_{i+1}$$
$$y_i + x_i . 2^{-I} . d_i = y_{i+1}$$
$$z_i - d_i \tan^{-1}(2^{-i}) = z_{i+1}$$

III.CORDIC MODES

Two modes are there in CORDIC. The first one is Rotation mode [15], in this mode input vector value rotated by a specified angle. Second one is Vectoring mode; in this mode input vector rotates to x-axis.

A. *ROTATION MODE*

In this mode angle accumulator is set to some preferred rotation angle. Here rotation angle serves as an argument for which we intend to find the trigonometric, hyperbolic or some other transcendental values. Angle accumulator will take care about rotation decision. The decision made at each iteration and evaluated.

CORDIC equations in rotation mode are:

$$x_i - y_i . 2^{-I} . d_i = x_{i+1}$$
$$y_i + x_i . 2^{-I} . d_i = y_{i+1}$$
$$z_i - d_i \tan^{-1}(2^{-i}) = z_{i+1}$$

Here,

$$d_i = -1 \text{ if } z_i < 0 \; +1, \text{ else}$$

n iterations after it produces following results:

$$z_n = 0$$
$$[y_0 \cos z_0 + x_0 \sin z_0] An = Y_n$$
$$[x_0 \cos z_0 - y_0 \sin z_0] An = X_n$$

B. *VECTORING MODE*

In vectoring mode input vector rotates through any angle towards *x* axis with resultant vector. Scaled magnitude and rotation angle and is the resultant for vectoring operation. The residual vector at each rotation it reduce the *y* component. *y* component sign residual used for determining the next rotation direction . Initially angle accumulator is set to zero, it continuous the traversed angle up to the final iterations.

A CORDIC equation in vectoring mode follows:

$$x_{i+1} = x_i - y_i . d_i . 2^{-i}$$
$$y_{i+1} = y_i + x_i . d_i . 2^{-i}$$
$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i})$$

Here,

$$di = +1 \text{ if } yi < 0 \; -1, \text{ else}$$

n iterations after it produces following results:

$$x_n = A_n (x_0^2 + y_0^2)^{1/2}$$
$$z_n = \tan^{-1}(y_0 / x_0) + Z_0$$
$$y_n = 0$$

IV. CORDIC ARCHITECTURES

CORDIC algorithm contains different architectures for mapping into hardware. Normally these are classified into folded and unfolded (figure 1). These operations done by comprehension of 3 iterative equations [14]. First method architecture of folded implemented by copying every difference equations of CORDIC into time multiplexing and hardware. In a single functional unit all these operations are done, so it will provide trading area for speed [16]. It can be classified into two types; those are word-serial and bit-serial architectures. It will check the functional unit implementation logic for the every one bit or word of CORDIC in each iteration.

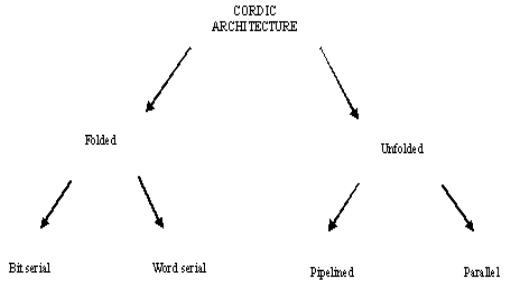

Fig: 1 Nomenclature of CORDIC

A. FOLDED WORD SERIAL ARCHITECTURE

This is done by duplicate the three difference equations in each. This is also called as iterative bit-parallel design [2, 17], the hardware as shown in figure 2. Each block contains a shift unit, subtraction-adder unit block, and one register used for output buffering. Initial values are given to register by the multiplexer for first level calculation. Here z-branch of MSB stored value determines the operation mode for the adder-subtraction unit. *X* and *y* outlet Signals passes to block of shift unit and finally subtracted or added to un-shifted signal value in reverse path.

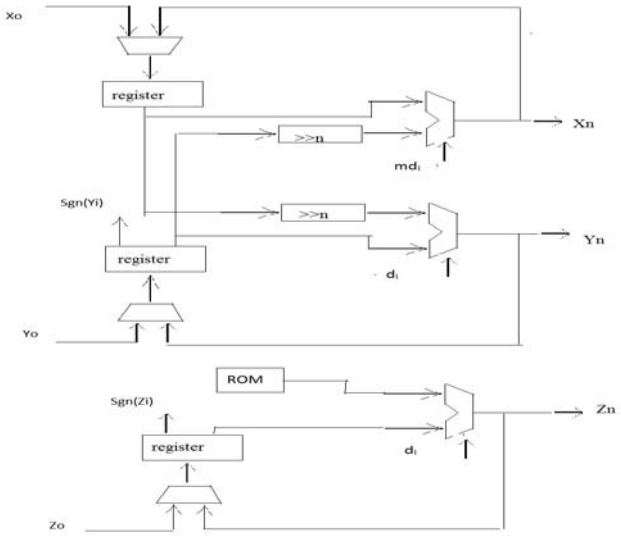

Fig:2 Folded Word Serial design

The *z* branch mathematically mixing the registers values, lookup table passes these values, then each number of operations the address value is changed. After n operations output is passes again to register block before primary values are fed in again and these final value can be access as output. Normal FSM needs to control the addressing of the constant values and multiplexers. All the initial values are given hardwired in a word wide manner when it is implemented in FPGA. Both subtracted and adder component are passed out separately. Angle accumulator controls the multiplexer. Routing signals is required to find distinguish between subtraction and addition. For changing the shift distance value shift operations are used, with the no of iterations even it require maximum speed and also high fan-in value for the application [3, 4]. Shifters not suitable for FPGA architectures because it wants several layers of logic. Then the resultant shows for large number of logic cells the design is slow.

B. *FOLDED BIT SERIAL DESIGN*

In engineering applications we are facing some problems like algebraic and nonlinear equations repeated evaluation in the field of signal processing areas and robotics, graphics in engineering. Complicated equation calculation will take huge time to calculate in software, even though we are using co-processors, and also contain large number transcendental and of nonlinear functions. Two designs of iterative bit-parallel and unrolled shows draw bag in path delays and complexity these are placed between single stages, these are large number of cross connections. Now we are going for bit-serial iterative architecture to reduce the complexity of

the design. In this process one bit is processed. Then one bit-wide data paths [1] are comes from cross connections. The modified bit serial design logic and interconnect work at high clock rate than bit parallel design.

This design model contain 3 shift registers, 3 bit serial subtraction- adder units, and serial ROM. word width length equal to shift register. Also it contain multiplexer block to select the shift registers taps, this is used for a cross terms of right shifted. This architecture is shown in figure 3. Here for n iterations w clock periods are needed. Precision adder is w here. It initializes the x, y and z registers with w clock periods for a load multiplexers. Once data is loaded, then the data is shifted to serial subtraction- adder block and finally return to register in left end. W clock periods require for each and every iteration to return register result value.

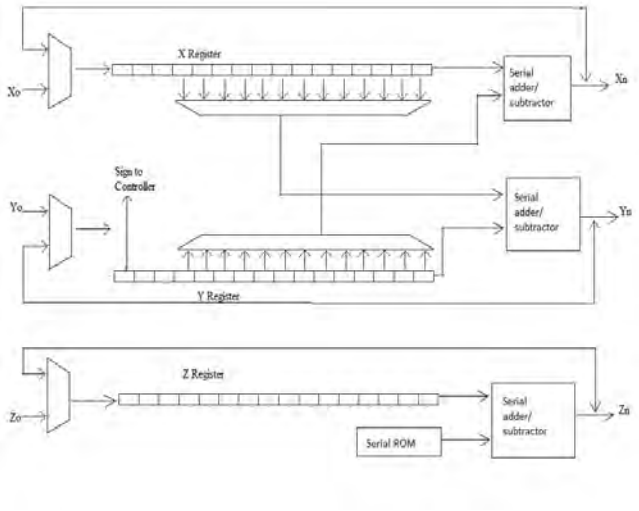

Fig:3 Folded Bit Serial Design

C. *UNFOLDED PARALLEL ARCHITECTURE*

The CORDIC processor discussed above is iterative algorithm, it means processor need to perform n iterations at the given data rate. This is an unfolded operation [17, 18] it means always performs the same iteration at n times processing elements. It shows in figure 4. This will result the value in two simplifications. First one is fixed shifts at shifters; by using wiring we can implement it. Another one based on lookup table. Here angle accumulator distributed the LUT values as constants to each adder; this is chain process in accumulator angle. Instead of using requiring storage space we are using constants, those are hardwired. The whole CORDIC processor can be modified into grouping of subtraction-adder unit in interconnected manner. Now we do not require registers, building of this unrolled processor stringently combinatorial. Finally resulting circuit delay should be substantial; now processing time is decreased compare to iterative circuit.

Mostly and particularly in FPGA do not make any sense of using combinatorial large circuit what we required. This design of unrolled processor can be easily pipelined [19].this is done by in-between the subtraction- adder unit block we are inserting registers. More FPGAs already contain registers in each logic cell, so this pipeline registers process do not increase the hardware cost due to already presented registers in the logic block.

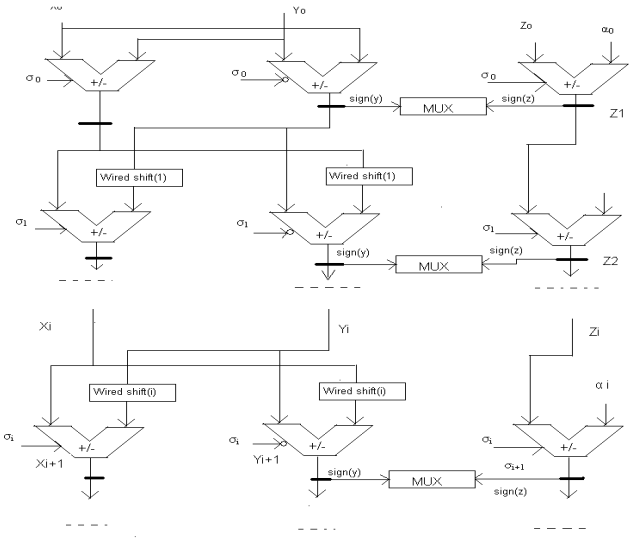

Fig:4 Unfolded Parallel Architecture

D. *PARALLEL AND PIPELINED ADDERS*

Pipelining concept mostly used in signal processing applications. If you consider simple example 128-bit adder constructed by using four consecutive 32-bit adders. This parallel adder shown in Figure 5 [20]. Here circuit computation time is almost equal to an 4.T, here T indicated as computation time of each 32-bit adder, now maximum sample rate will be equal to 1/(4.T) of input operands x and y. The pipelined circuit for corresponding data is shown in Figure 6. In this one register placed between the computations resources. These resources are allocated to consecutive cycles, in this way new addition is started when it completes the first cycle of the preceding addition. It is done T seconds. Now latency value is equal to 4.T. Then sample rate of 1/(4.T) is changed to 1/T.

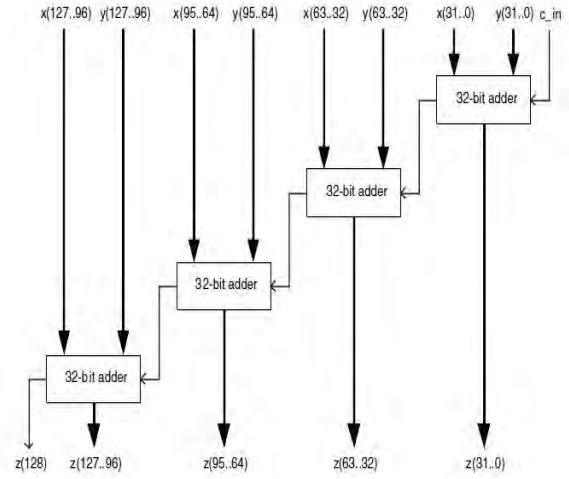

Fig: 5 Parallel Adder

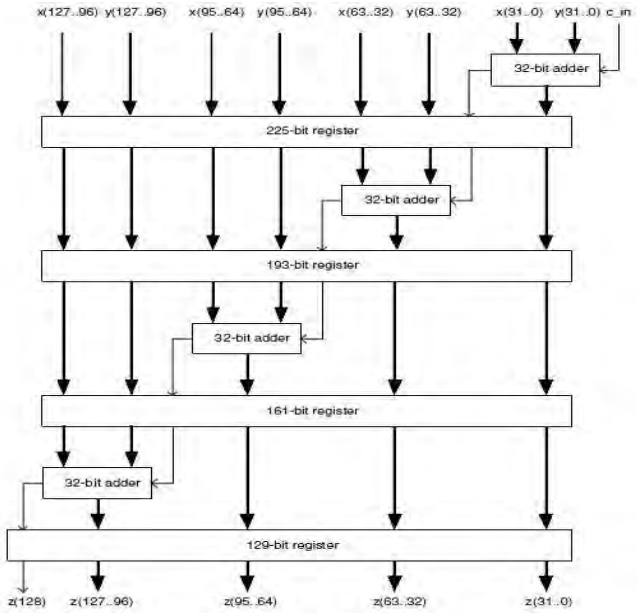

Fig:6 Pipelined Adder

Normally basic cell FPGA contain flip flops so pipeline registers insertion do not have necessity to increase the value of total cost, by using basic cells we are computed these values. The flip-flops in pipeline registers are not used in the version of Non-pipelined.

V.CONCLUSION

The CORDIC splendor is prospective for united solution for hefty set of tasks concerning the computation of multiplication, logarithmic and square-root, division, assessment of transcendental functions and trigonometric, elucidation of linear systems. CORDIC algorithm can be implemented via simple hardware through continual shift-add operations. Computation latency endures to be fore most disadvantage of CORDIC algorithm, ever since we do not have proficient algorithms for its parallel implementation. With enrichment of its throughput and diminution of its latency it is anticipated that CORDIC would be positive for various real-time and high-speed applications. From fast decade, numerous architectures and algorithms have been urbanized to speed up CORDIC algorithm by plummeting its iteration counts by implementation of pipelined architecture. Further way to exploit CORDIC competently, is to go for Redundant Arithmetic.

REFERENCES

[1] Volder J. E., “The CORDIC trigonometric computing technique”, IRE Trans. Electronic Computing, Volume EC-8, pp 330 - 334, 1959.
[2] Walther J. S., “A unified algorithm for elementary functions,” in Proceedings of the AFIPS Spring Joint Computer Conference, pp. 379–385, May 1971.
[3] Walther J. S., “The story of Unified CORDIC,” Journal of VLSI Signal Processing, vol. 25, no. 2, pp. 107–112, 2000.
[4] Andraka R, “A survey of CORDIC algorithms for FPGA based computers,” FPGA ’98, ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp 191-200.
[5] De Lange A. A. J., Van der Hoeven A. J., Deprettere E. F., and Bu J., “Optimal floating-point pipeline CMOS CORDIC processor,” Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS ’88), vol. 3, pp. 2043–2047, June 1988.
[6] Despain A. M., “Fourier transform computers using CORDIC iterations,” IEEE Transactions on Computers, vol. 23, no. 10, pp. 993–1001, 1974.
[7] Ahmed H. M., Delosme J. M., and Morf M., “Highly concurrent computing structures for matrix arithmetic and signal processing,” Computer, vol. 15, no. 1, pp. 65–82, 1982.
[8] Cavallaro J. R. and Luk F. T., “CORDIC arithmetic for an SVD processor,” Journal of Parallel and Distributed Computing, vol. 5, no. 3, pp. 271–290, 1988.

[9] Lee J. A. and Lang T., “SVD by constant-factor-redundant- CORDIC,” Proceedings of the 10th IEEE Symposium on Computer Arithmetic, pp. 264–271, June 1991.
[10] Meyer A., Watzel R., Meyer U., and Foo S., “A parallel CORDIC architecture dedicated to compute the Gaussian potential function in neural networks,” Engineering Applications of Artificial Intelligence, vol. 16, no. 7-8, pp. 595– 605, 2003.
[11] Kang C. Y. and Swartzlander E. E., “Digit-pipelined direct digital frequency synthesis based on differential CORDIC,” IEEE Transactions on Circuits and Systems I, vol. 53, no. 5, pp. 1035–1044, 2006.
[12] Antelo E., Lang T. and Bruguera J. D., “Very-High Radix CORDIC Rotation Based on Selection by Rounding,” Journal of VLSI Signal Processing, Kluwer Academic Publishers, Netherlands, Vol.25, 141.153, 2000.
[13] Delosme M. J.,Lau C. Y. and Hsiao S. F., “Redundant Constant-Factor Implementation of Multi-Dimensional CORDIC and Its Application to Complex SVD,” Journal of VLSI Signal Processing, Kluwer Academic Publishers, Netherlands, Volume 25, pp 155.166, 2000.
[14] Choi J. H., Kwak J. H. and Swartzlander, Journal of VLSI Signal Processing, Kluwer Academic Publishers, Netherlands, Volume 25, 2000.
[15] Meggitt J. E., “Pseudo division and pseudo multiplication processes” IBM Journal, vol. 6, no. 2, pp. 210–226, 1962.
[16] Deprettere E., Dewilde P., and Udo R., "Pipelined CORDIC Architecture for Fast VLSI Filtering and Array Processing," Proc. ICASSP'84, 1984, pp. 41.A.6.1- 41.A.6.4.
[17] Lin C. H. and Wu A. Y., “Algorithm and Architecture for High-Performance Vector Rotational DSP Applications,” Regular IEEE Transactions: Circuits and Systems I, Volume 52, pp 2385- 2398, November 2005.
[18] Erecegovac M. D. and Lang T., Digital Arithmetic, Elsevier, Amsterdam, the Netherlands, 2004.
[19] Hu Y. H., “Pipelined CORDIC architecture for the implementation of rotational based algorithm,” in Proceedings of the International Symposium on VLSI Technology, Systems and Applications, p. 259, May 1985.
[20] De Lange A. A., van der Hoeven A. J., Deprettere E. F., and Bu J., An optimal floating-point pipeline CMOS CORDIC Processor," IEEE ISCAS'88, pp. 2043-47, 1988.
[21] Burhan Khurshid “FPGA Implementation of a High Speed Multistage Pipelined Adder Based CORDIC Structure for Large Operand Word Lengths” :International Journal of Computer Science and Telecommunications [Volume 3, Issue 5, May 2012]