IMPLEMENTATION OF SCALING FREE CORDIC AND ITS APPLICATION

M.R.Maanasa
(M.Tech) VLSI &Embedded systems,
Department of E.C.E,DBIT,Bangalore
mnr.khadri@gmail.com

N.Asha
Asst.Professor,
Department of E.C.E,DBIT,Bangalore
ashagowda.mail@gmail.com

ABSTRACT- The CORDIC method is the most versatile of all the algorithms that can be used to compute elementary functions. The way of computing sine and cosine of angles, involves in multiplication free, thus power efficient and which results in regular architecture and less complicated routing, consequently less area, simultaneously lead to high throughput. For that purpose CORDIC seems to be a best solution. CORDIC offers a unified iterative formulation to efficiently evaluate the trigonometric function using rotation operation. This can be achieved by rotating a vector by an angle to form a newly rotated vector, introducing scaling factor. In order to achieve simplicity of hardware realization and benefits over basic CORDIC, CORDIC is designed by eliminating scaling factor. Here, scaling free CORDIC is proposed in order to compare with basic CORDIC. The proposed Scaling free is integrated with DFS (Direct Frequency Synthesizer) architecture so as to generate sine and cosine waveform from the values generated in each iterations. The proposed design computes sine and cosine of angles and thus maintains the regularity of basic CORDIC structure. The generated sine and cosine waveforms called DDS (direct digital synthesis) thus has wide applications in the field of RF signal processing, satellite communications etc.. The CORDIC structure is implemented in Xilinx FPGA.

Keywords:Coordinate rotation digital computer (CORDIC), Digital frequency synthesis (DFS)

1. INTRODUCTION

CORDIC (for COordinate Rotation Digital Computer), is a simple and efficient algorithm to calculate trigonometric functions, also known as the digit-by-digit method or Volder's algorithm. It is commonly used when no hardware multiplier is available (e.g., simple microcontrollers and FPGAs) as the only operations it requires are addition, subtraction, bit shift. It comprises a special serial arithmetic unit having three shift registers, three adders/subtractors, and special interconnections.

CORDIC computes a number of functions including sine and cosine of angles. Since, CORDIC architecture is multiplication free(adder/subtractor as the main computational block), it has been used as a basic unit of computation in the implementation of different algorithms like discrete cosine transforms (DCT), discrete Hartley transform (DHT), fast Fourier transform (FFT), etc. The advantage of this algorithm is that it uses minimal hardware (adder and shift) for computation of all trigonometric and other function values and so performance is high. Thus, almost all scientific calculators use CORDIC algorithm in their calculations. This algorithm was implemented when Jack Volder developed CORDIC algorithm in 1959 to calculate the trigonometric relationships. Through careful selection of the scale factor, Arc Tangent Radix, and initial conditions, Volder was able to develop a family of iterative equations that only required shifts and adds to calculate the trigonometric functions in a deterministic number of operations.

In regular CORDIC VLSI structure, ROM is used to store the precomputed values of arctangents. However, ROM based design is not preferred because ROM has slow speed (ROM access time) and more power consumption. Ever since the CORDIC algorithm was developed, there have been many attempts to solve the drawbacks, which are the latency of computation and the dependency of a scaling factor. Scaling free CORDIC is one such relying on Taylor series expansion, decreasing the required number of iterations.

However, the scaling factor is one of the major drawbacks with the Basic CORDIC algorithm, in particular when optimizations or extensions to the algorithm are considered. The motivation for including this into analysis is that it performs particularly well for evaluation of cosine and sine and is therefore useful as a reference. The scaling free CORDIC algorithm removes the need of scaling factor and hereby making it possible to skip some of the iterations if they are not needed, there by overcoming latency problem of basic CORDIC. The values obtained in each iteration of scaling free CORDIC is fed as input to the digital frequency synthesizer. There by generating waveforms according to the angles generated in each iterations. CORDIC algorithm is one of the technique to generate carrier wave in communication system to perform modulation and demodulation. Digital Frequency Synthesizers also termed as Direct digital synthesizers (DDS), are a class of frequency synthesizers in digital domain, which generate waveforms of some frequencies. Sometimes also called numerically controlled oscillators

(NCO), these generate waveforms like sine, cosine, triangular, square or rectangular, saw tooth, etc. These have wide applications in satellite communication systems, RF signal processing, etc. DDS offers many advantages over analog oscillators such as extremely precise tuning resolution of the output frequency, fast hopping of phase which reduces phase related errors, remotely controllable, better match of quadrature outputs when required, etc.

2. CORDIC METHODOLOGY

CORDIC works by rotating the coordinate system through constant angles until the angle is reduced to zero. So with this principle the given angle is changed each time to reduce to zero. Here addition, subtraction and shift operations are used to calculate the function values. All the trigonometric functions can be computed or derived from functions using vector rotations. The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shift and add operations. The algorithm is derived using the general rotation transform.

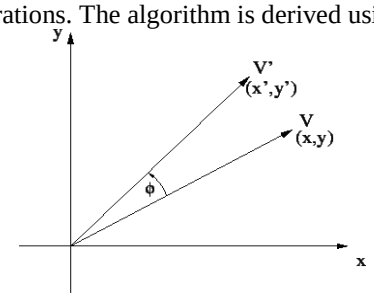


Fig.2.1 Rotation of a vector V by an angle φ

If a vector V with components x and y is rotated with an angle φ ,the new vector v' with components (x',y') formed is as shown in Fig.2.1. The objective is to find the new vector coordinates (x', y'), which are identical to the cos and sin values which can be derived using the general rotation transform by and the equations for x and y coordinate is given by

$$x_{i+1} = K_i(x_i - d_i y_i 2^{-i}) \quad 2.1$$

$$y_{i+1} = K_i(x_i + d_i y_i 2^{-i}) \quad 2.2$$

where d_i is the direction of rotation , d_i is 1, if $z_i \geq 0$ else (-1). From equation 2.1 and 2.2, each iteration is multiplied with K_i which is denoted as scaling factor. Removing the scaling factor yields an iterative shift-and-add algorithm for vector rotation. However, the scaling factor is still one of the major drawbacks with the CORDIC algorithm, in particular when optimizations or extensions to the algorithm are considered. The scaling-factor K_i, after sufficiently large number of iterations converges to a constant value 0.607. A third iterative component shown in equation 2.3 is needed to keep track of the rotations of the angle. The rotations are accumulated with the help of a lookup table holding the values of $\alpha_i = \text{atan}(2^{-i})$.

$$z_{i+1} = z_i - a\tan(2^{-i}) \quad 2.3$$

atan (2^{-i}) in equation 2.3 can be written as α_i , Parameter α_i is angle by which a vector is rotated in i^{th} step. These are called Arctangent values, which are constant and they are stored in ROM. From the above equations CORDIC can be modeled as shown in below figure 2.2.

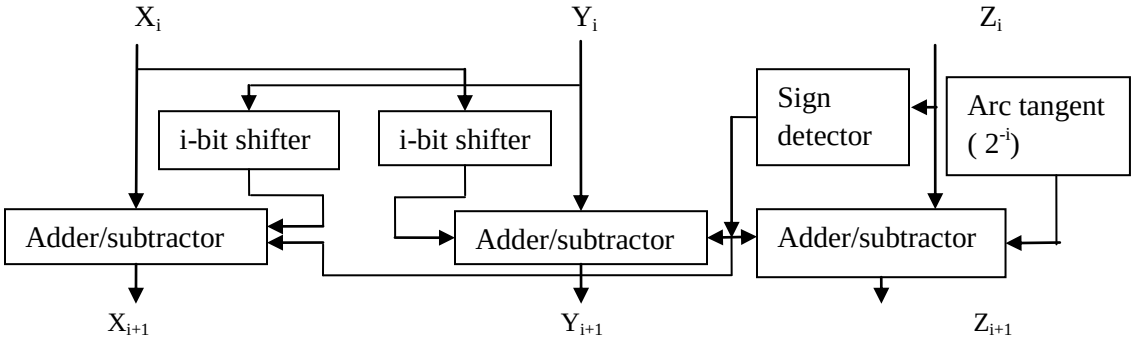


Fig.2.2 Basic architecture of CORDIC processor

For the iteration to go from i^{th} stage to $(i+1)^{th}$ stage, the sign of Zi has to be predetermined. Adder/subtractor is the only computational unit in *Z*- datapath, latency of this architecture is determined by the latency of the adder/subtractor module. Depending on the sign of the input angle, the adder/subtractor adds or subtracts. The shifter shifts according to i^{th} value.

3. IMPLEMENTATION OF SCALING FREE CORDIC

Scaling free can also be implemented with only shift and add operations. The scaling free algorithm does only perform rotations in one direction. The accumulation of the angle for the scaling free algorithm does not need a lookup table. This means a decrease in the number of iterations, since it can skip unnecessary iterations and also elimination of scale factor. Here the scaling free CORDIC itself is going to generate the arctangents through micro rotations when compared to basic CORDIC. The starting point for the scaling free algorithm uses a Taylor series expansion of cos x and sin x. The proposed design is based on the following key ideas: 1) Taylor series expansion of sine and cosine functions are used to avoid scaling operation and 2) suggest a generalized sequence of micro-rotation. The angle is partitioned into smaller rotations like the basic CORDIC, and if the rotations are small enough then one scaling free rotation can be approximated as which means that Scaling-free CORDIC was the first attempt to completely dispose of the scale-factor. Here, the sine and cosine functions were approximated using the first order of Taylor series to equation 3.1 as

$$\cos\alpha_i = 1 - \frac{\alpha_i^{\,2}}{2!} = 1 - 2^{-(2i+1)}$$
$$\sin\alpha_i = \alpha_i = 2^{-i} \qquad 3.1$$

Unlike the conventional CORDIC algorithm, in the scaling free CORDIC algorithm, the final target angle is achieved by rotating the vector in one direction only. This means that the final target angle is approximated as a pure summation of the elementary angles. Design of CORDIC processor is divided in two main parts, the Arithmetic Calculation Modules and Shift Calculation Modules.

3.1 Arithmetic Calculation Modules

The expansion of the Taylor series approximation, is in the below equation 3.3.

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = \begin{bmatrix} (1-(2!)^{-1}\alpha^2{}_i) & -(\alpha_i - 2^{-3}\alpha^3{}_i) \\ \alpha_i - 2^{-3}\alpha^3{}_i & (1-(2!)^{-1}\alpha^2{}_i) \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \qquad 3.2$$

Assuming $\alpha_i = 2^{-si}$ and approximation of factorial, these are expressions for Micro-Rotations Using Taylor Series Approximation and Factorial Approximation. The arithmetic calculation modules were used in implementing the above equation 3.2 as shown in Fig.3.1. 16-bits register is used at the end of the module for synchronization purpose. This module was implemented by using fixed point format. 16-bits used for the xi and y_i input. The value of s_i obtained from the shift calculation module. The output of this module will be used as input to the next iteration, and the iteration continues until the counter is reset to zero. This is used to find the x and y coordinates in the coordinate calculation unit. Here Taylor series is used for coordinate calculation.

3.2 Shift calculation module

The fixed point format of the elementary angle (α_s) has one bit set and represented by $\alpha_{si} = 2^{-si}$. The first one bit number of any input string counting from the Most Significant Bit (MSB), M is used for further calculation of shift iteration (s_i), for a fix word-length (N). S_i of the elementary angle was given by equation 3.3 as

$$S_i = N\text{-}M \qquad 3.3$$

In this, the word-length is fixed to 16. The module's input is the angle to be rotated, theta (θ_i). The elementary angle (α) is corresponding to the basic shift (s). Basic shift was the first elementary angle for rotation α. The expression for the basic shift is as stated in equation 3.4 as

$$\text{basic}-\text{shift}, s = \left\lfloor \frac{b - \log_2(n+1)!}{n+1} \right\rfloor \qquad 3.4$$

Where b is the word length and n is the total number of iterations. In shift calculation module, multiple iteration of shift is performed. Shift calculation module is used to get the shift index (s_i) parameter. The operation begins to get the first 1 bit from the MSB (M). The code for generating micro rotation sequence is as shown below:
Input: angle to be rotated(θ_i)
Begin
M= mostsignificant-1 location of θ_i
α= 0.25 radians
shift s_i=2 and $\theta_{i+1}= \theta_i-\alpha$
else
shift s_i=16-M
$\theta_{i+1}= \theta_i$ with $\theta_i[M]$='0'
end

Fig.3.1 shows the architecture of the proposed scaling free CORDIC processor. The block diagram for the proposed CORDIC architecture can be explained as; it makes use of the same stage for all the iterations for the coordinate calculations, as well as for the generation of shift values. The data to be found out is saved upon mux selection and is fed to stage.

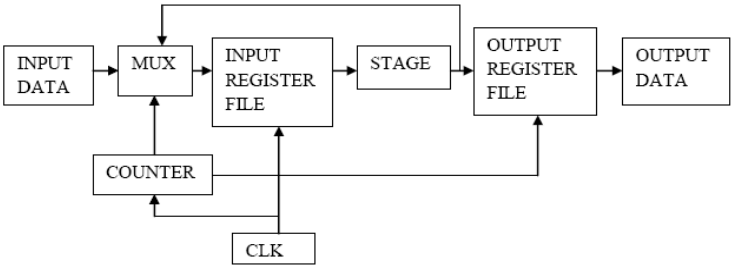


Fig.3.1 Architecture of proposed scaling free cordic

The structure of stage block is as shown in Fig.3.2. Here x and y coordinate is calculated from the coordinate calculation unit, micro rotation sequence generation block generates new theta and shifted value required for the next iteration. The output of stage is fed as input to the next iteration upon mux and counter and the iteration continues until the counter is reset to zero. The expiry of the counter indicates the completion of a CORDIC operation.

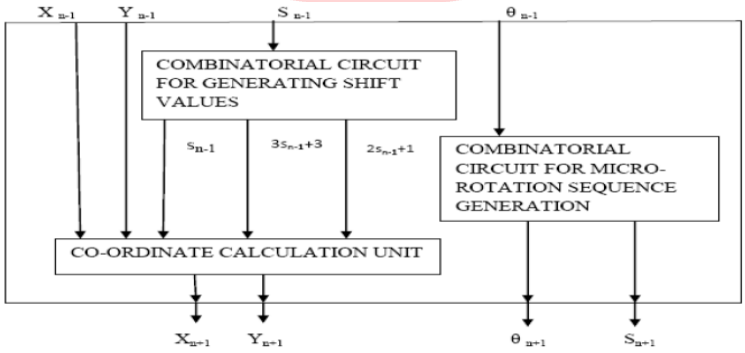


Fig.3.2 Block diagram of each stage

The combinatorial circuit for the evaluation of desired shift values is shown in Fig.3.3

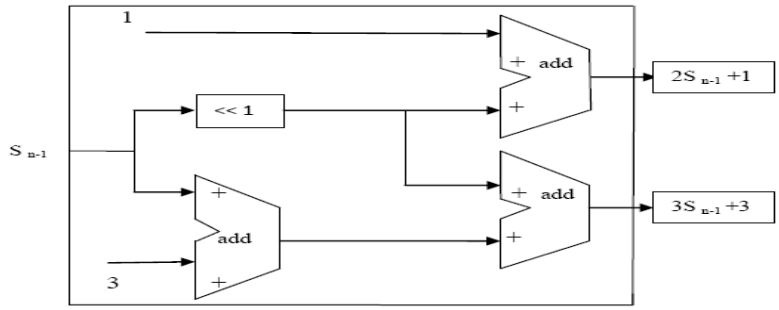


Fig.3.3 Circuit for generating the shift values

The combinatorial circuit for generating the micro-rotation sequence is shown in Fig.3.4.

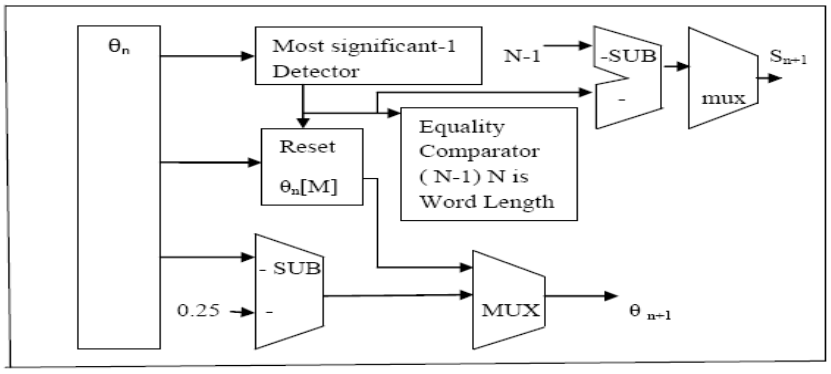


Fig.3.4. Micro-rotation sequence generation

3.3 Micro-Rotation Selection

The proposed micro-rotation sequence in Fig.3.4 can be explained as, multiple iterations of basic-shift are performed, followed by non-repetitive unidirectional iterations of the micro-rotations corresponding to other shift indices, to minimize the number of iterations .

A. Micro-Rotation Sequence organization

In the proposed scheme, the rotation angle "θ" is represented as

$$\theta = n_1 . \alpha_s + \sum_{i=0} \alpha_{si} \qquad 3.5$$

where $n=n_1+n_2$, α_s is the elementary angle corresponding to the basic-shift, α_{si} are elementary angles for other shifts, n_1 and n_2 are non-negative integers and n represents the total number of iterations. If any micro-rotation of angle α_s is not used then n_1 is zero, and $n_2=n$. On the other hand, if the desired angle of rotation "θ" is a multiple of α_s then n_2 is zero and $n_1=n$.

B. Defining the Elementary Angles

The elementary angles α_s and α_{si} are given by

$\alpha_{s=} 2^{-s}$, $\alpha_{si}=2^{-s}$ 3.6

where, s is the basic-shift and $s_i>s$ is the shift for ith iteration. For basic-shift=2, $\alpha_s = 7\pi/88$ and for basic-shift 3, $\alpha_s = 7\pi/176$.

C. Generalized Micro-Rotation Sequence Identification

The micro-rotations are identified depending on the bit representation of the desired rotation angle in radix-2 system using most-significant-1 detector. For this, the maximum rotation angle is restricted to $\pi/4$ radians as the entire coordinate space $[0,2\pi]$ can be mapped to the $[0, \pi/4]$ using octant symmetry of sine and cosine functions. If the most-significant-1 location (M) of the rotation angle "θ" is smaller than the basic-shift "s", elementary angle of the

basic-shift would be used for the CORDIC iteration. For a fixed word-length of N-bit, the shift s_i for the elementary angle is given by S_i=N-M.

This is how the scaling free CORDIC works. Here the values obtained in each iteration are fed as input to the direct digital synthesizer in terms of frequency control word. There by generating waveforms according to the angles generated in each iterations.

3.4 Digital Frequency Synthesizers

Simplified form of DFS is shown in the below Fig.3.5. It consists of a phase accumulator and a phase to amplitude converter (conventionally a sine ROM). The phase accumulator consists of a j bit frequency register, which stores a digital phase increment word followed by a j bit full adder and a phase register. The digital phase increment input word is entered in the frequency register. This data is added to the data previously held in the phase register at each clock pulse. The phase increment word represents a phase angle step that is added to the previous value at each $(1/f_{clk})$ second to produce a linearly increasing phase value as shown in the Fig.3.5 (b).The phase is generated by modulo 2^j overflowing property of a phase accumulator. The rate of overflow is consider as the output frequency, which is expressed as

$$f_{out} = \frac{\Delta P f_{clk}}{2^j} \forall f_{out} \leq \frac{f_{clk}}{2} \qquad 3.7$$

where ΔP is the phase increment word, j is the number of phase accumulator bits, f_{out} is the output frequency and f_{clk} is the clock frequency. The constraint in the above equation 3.11 comes from the sampling theorem. Frequency resolution is found by setting ΔP =1 as the phase increment word is an integer, as

$$\Delta f = \frac{f_{clk}}{2^j} \qquad 3.8$$

Digital phase information is converted into the values of a sine wave from the ROM, which is the look-up table.

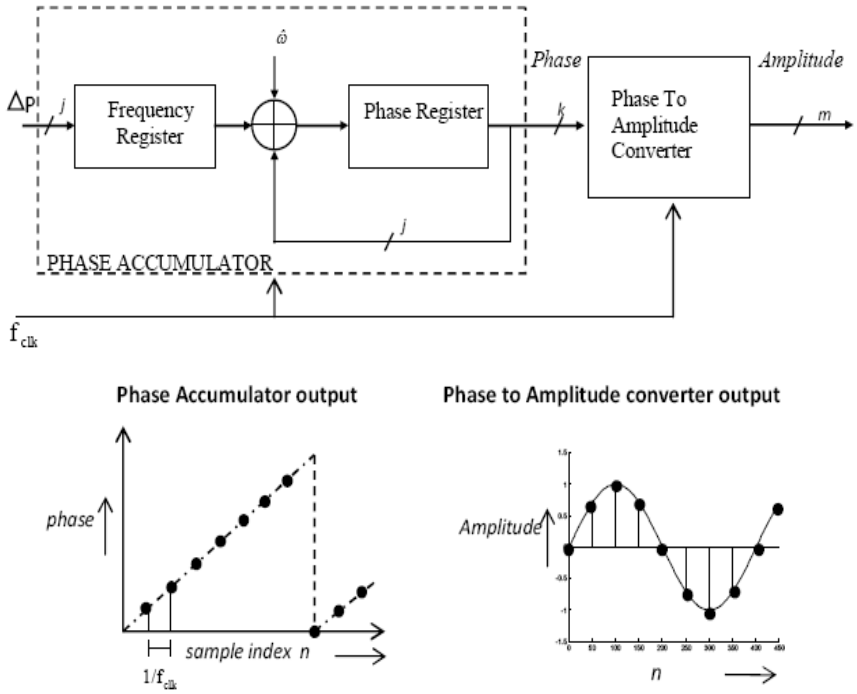


Fig 3.5 DFS block diagram and wave shapes

3.4.1 Blocks of Digital Frequency Synthesizer

A. Phase Accumulator

A clock with frequency f_{clk} is the synthesizer's only time reference. The phase accumulator's output is a ramp value, as it overflows to 0 periodically. For an N-bit accumulator, the frequency of the ramp is given by

$$f_{out} = f_{clk} X \frac{\text{frequency control word}}{2^n} \quad 3.9$$

Every value at the output of the phase accumulator is converted to approximated sine amplitude by a phase-to-sine amplitude converter.

B. Phase to Amplitude Converter

The spectral purity of the DFS is estimated by the values stored in the sine table ROM. Hence can increase the resolution of the ROM. But as ROM storage increases lower the speed, increased power consumption and greatly increased costs. By storing only π/2 radians of sine wave information compression can be achieved and to generate the ROM samples for the full range of 2π by exploiting the quarter wave symmetry of the sine function.

One of the approaches to the phase-to-sine amplitude mapping is the CORDIC algorithm, which is an iterative computation method. But there is increased circuit complexity, cost and distortions that will be generated, when the methods of memory compression are employed. Technique to store only π / 2 radians of sine information and to generate the sine look-up table samples for the full range of 2π is as explained in the exploitation of sine function symmetry.

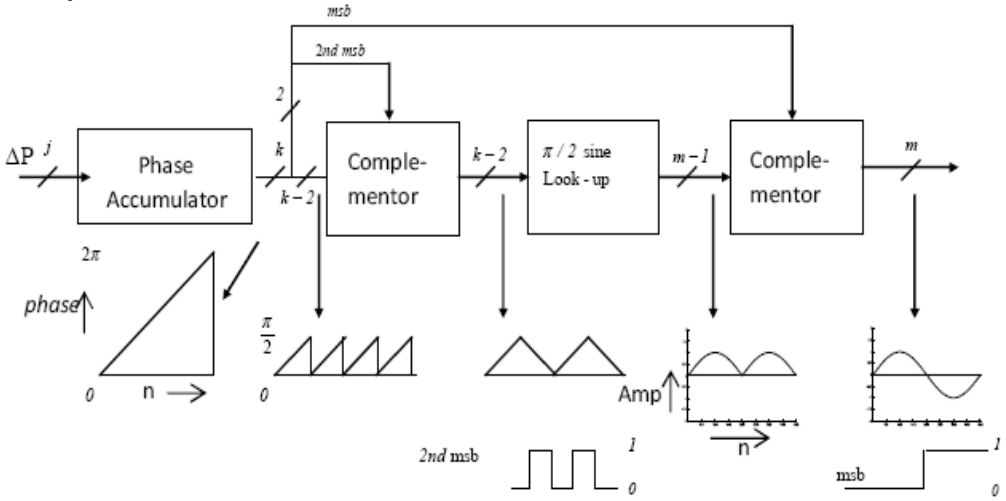


Fig.3.6 Detailed diagram of DFS

3.4.2 Exploitation of Sine Function Symmetry

Technique to store only π / 2 radians of sine information and to generate the sine look-up table samples for the full range of 2π quarter-wave symmetry of the sine function is used. The decrease in the look-up table capacity is paid for by the additional logic necessary to generate the complements of the accumulator and the look-up table output, as shown in Fig.3.6. The two Most Significant Bit (MSB)s are used to decode the quadrant, remaining k-2 bits are used to address a one-quadrant sine look-up table. MSB determines whether the amplitude is increasing or decreasing. The accumulator output is used "as is" for the first and the third quadrants. The bits must be complemented so that the slope of the saw-tooth is inverted for the second and fourth quadrant. The sampled waveform at the output of the look-up table is a full wave rectified version of the desired sine wave as shown in Fig.3.6. The final output sine wave is then generated by multiplying the full wave rectified version by -1, when the phase is between π and 2π.

3.4.3 Concept of the Architecture Used

Instead of a ROM LUT, a hardware-optimized phase-to-sine amplitude converter used to approximate the first quadrant of the sine function with eight equal-length piecewise linear segments as shown in Table 3.1. The main goal is to maintain low system complexity and reduce power consumption and chip area requirements. The second

aim is to achieve a specified spectral purity, where spectral purity is defined as the ratio of the power in the desired frequency to the power in the greatest harmonic, across tuning bandwidth of the synthesizer. Spectral purity is an essential design parameter in communication systems for synthesizer, ensuring that undesired in-band signals remain below a given threshold and are not detected. In order to achieve the first goal, approximation of a sinusoid as a series of eight equal-lengths piecewise continuous linear segments s_i is done in equation $s_i(x)$, where

$$s_i(x) = m_i \times (x - \frac{i}{8}) + y_i \qquad 3.10$$

Where $i \in [0, 7]$ is the slope of each segment and is carefully selected to eliminate the requirement for multiplication by representing each one as a sum of at the most two powers of two. Precision of slope representation, i.e., the difference between the smallest and the largest powers of two used can restrict, by putting an upper bound on the adder's width. To reduce the control system circuitry costs equal length segments are selected. In order to obtain desired spectral purity, different sets of m_i and y_i coefficients are evaluated and the best one meeting the requirements is selected.

3.4.5 Description of the Architecture

Complete DFS architecture is shown in Fig.3.7, the coefficients are given in Table 3.1. The phase to sine amplitude converter block includes a 1's complement to exploit quarter wave symmetry. This architecture is significantly less complex. It does not include a ROM, no multipliers or squaring circuits are required.To simplify the control circuitry equal length segments are used. Only three integers need to be added and multiplexers. The phase accumulator is of 20 bits wide, truncated to 12 bits. The two MSBs are used for quadrant symmetry. Segment is identified by the next three bits. The remaining seven bits identify different sub-angles. The two upper multiplexers shift these remaining seven bits according to the slopes m_i , listed in Table 3.1.

Table 3.1 Linear segment coefficients

i	m_i	y_i
0	1+1/2	2/1024
1	1+1/2	191/1024
2	1+1/2	384/1024
3	1+1/8	552/1024
4	1	697/1024
5	1/2+1/4	819/1024
6	1/2	909/1024
7	1/8	971/1024

The notation {>>n} shown in Fig.3.7 signifies a right shift by n bits, means division by 2^n. The lower multiplexer selects the appropriate y_i approximation listed in the table. The output from the multiplexers is of 13 bits wide, to account for the whole dynamic range of possible values. The three-operand adder sums the multiplexer outputs together and rounds the result to 7 bits.11111e main advantage of this architecture is that it does not depend upon the extensive use of ROM, as is normally the case with other commonly available architectures. Hence, it fits into a very small area on the chip. Fine frequency and phase resolution can be achieved using DFS.

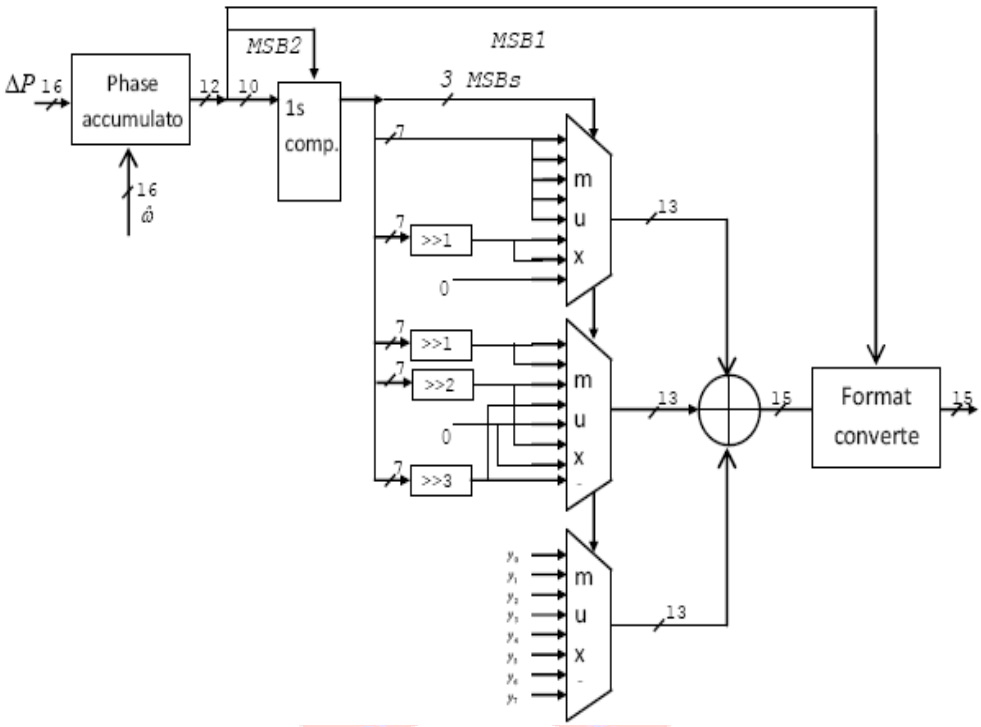
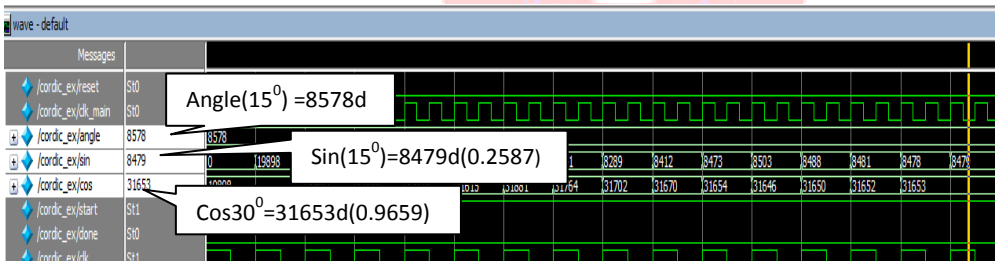


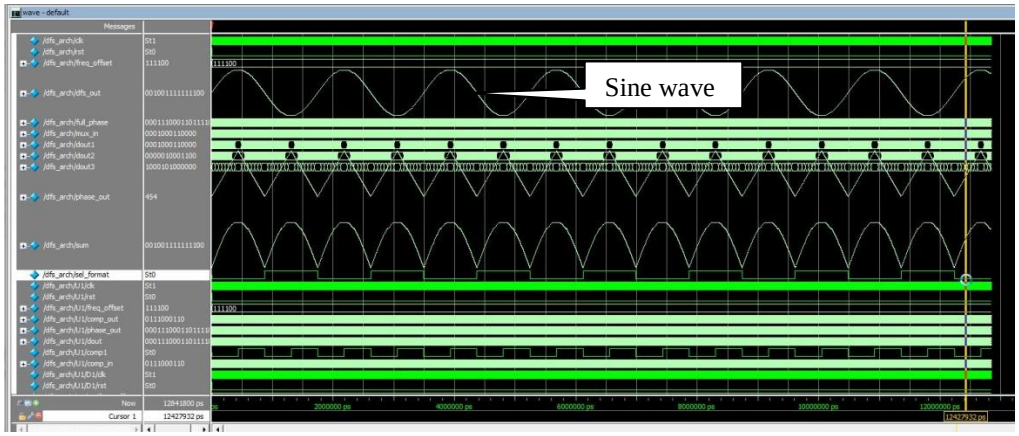
Fig.3.7 DFS architecture

4.RESULTS

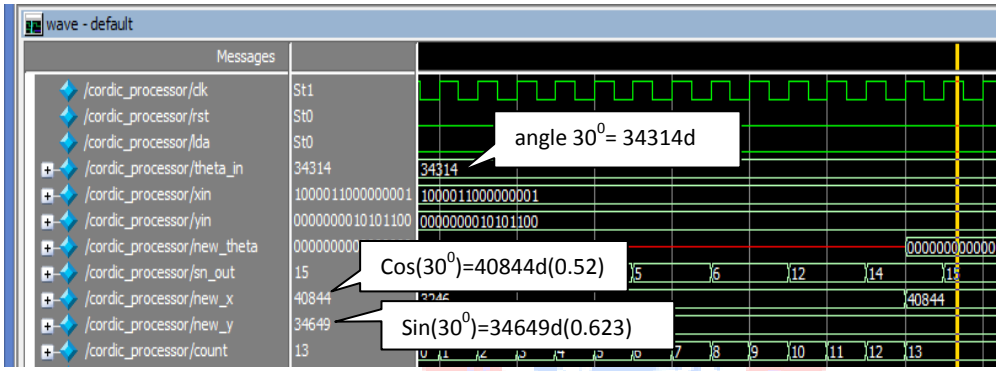
CORDIC has been implemented using FPGA hardware and by using verilog language. The synthesizer used here is Xilinx ISE and simulator used is Modelsim. The device used here is xc3s400-5pq208. Since verilog code is written, inputs and outputs are written in decimal format. Results of basic cordic is shown below.



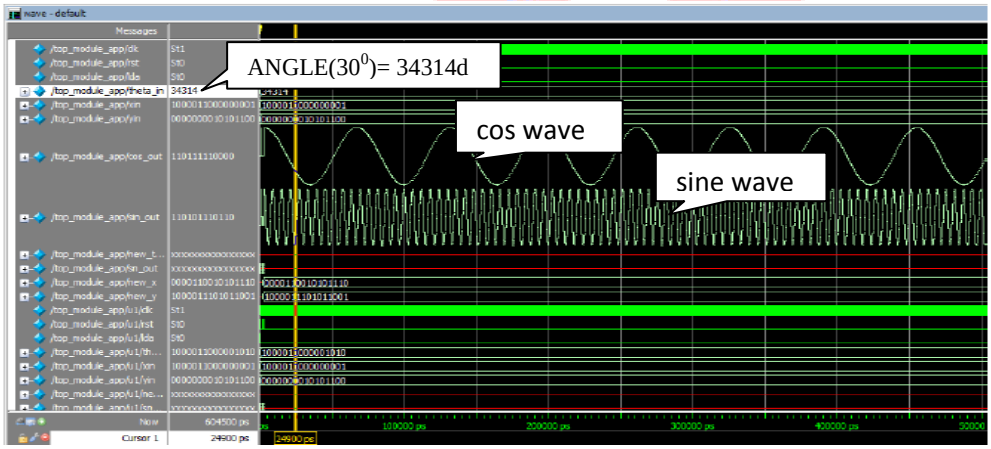
Results of DFS :



Results of scaling free CORDIC:



Results for integration of DFS in to scaling free CORDIC:



5. CONCLUSION

In this project a CORDIC module is designed and simulated using Xilinx ISE using Verilog as a synthesis tool. The output of the CORDIC core is analyzed and verified. The output values of basic cordic were found to be consistent with the actual values and also for scaling free with 0.2% error since it itself generates shift and theta values leading to less iterations. Scaling-free CORDIC algorithm provides an iterative calculation using only add and shift function for cosine and sine function calculation same as basic CORDIC. The elimination of redundant iterations using leading-one-bit technique improves the number of iteration compared to basic CORDIC. Less iteration offers faster

computation and optimal efficiency. The implementation of this scaling-free CORDIC will improved the performance of the application in term of efficiency and stability.

REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Transactions on Electronic Computers, vol. EC-8, pp. 330–334,Sept. 1959.
[2] J. S. Walther, "A unified algorithm for elementary functions," Joint Spring Computer Conference Proceedings, vol. 38, pp. 379–385, Jul. 1971.
[3] P. K. Meher, J. Valls, T. B. Juang, K. Sridharan and K. Maharatna, "50 Years of CORDIC: Algorithms, Architectures, and Applications," IEEE Transactions on Circuits and Systems—I: Regular Papers, Vol. 56(9),pp.1893-1907, Sept. 2009.
[4] H. Jeong, J. Kim, and W. K. Cho, "Low-power multiplierless DCT architecture using image data correlation," IEEE Transactions on Consumer Electronics, 50 (1), pp. 262– 26, Feb. 2004.
[5] Cheng-Ying Yu, Sau-Gee Chen, and Jen-Chuan Chih, "Efficient CORDIC Designs for Multi-Mode OFDM FFT," IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp. 1036-1039, May 2006.
[6] B. Das and S. Banerjee, "Unified CORDIC-based chip to realize DFT/DHT/DCT/DST," IEE Computers and Digital Techniques, Proceedings,Vol. 149(4), pp.121-127, Jul 2002.
[7] C.-C. Sun, S.-J. Ruan, B. Heyne and J. Goetze, "Low-power and high-quality Cordic-based Loeffler DCT for signal processing," IET Circuits, Devices & Systems,Vol.1(6), pp.453-461, Dec.2007.
[8] Jue-Hsuan Hsiao, Liang-Gee Chen, Tzi-Dar Chiueh, and Chun-Te Chen "High Throughput CORDIC-Based Systolic Array Design for the Discrete Cosine Transform," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5(3), June 1995.
[9] S. Wang, V. Piuri, Wartzlander, Jr. E. E., "Hybrid CORDIC algorithms," IEEE Transactions on Computers, Vol. 46(11), pp.1202-1207, Nov. 1997.
[10] T. B. Juang, S. F. Hsiao and M. Y. Tsai, "Para-CORDIC:Parallel CORDIC Rotation Algorithm," IEEE Transactions on Circuits and System I, vol. 51(8), pp. 1515–1524, Aug. 2004.
[11] Y. H. Hu, "CORDIC-based VLSI Architectures for Digital Signal Processing," IEEE Signal Processing Magazine, pp.16-35,Jul 1992.
[12] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling free adaptive CORDIC rotator algorithm and architecture," IEEE Transactions on Circuits and Systems for Video Technoogy, vol. 15(11), pp. 1463–1474, Nov. 2005.
[13] N. Takagi, T. Asada and S. Yajima, "Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation," IEEE Transactions on Computers, Vol. 40(9), pp.989-995, Sept. 1991.
[14] D. Timmermann, H. Hahn and B. J. Hosticka, "Low Latency Time CORDIC Algorithms," IEEE Transactions on Computers, Vol. 41(8), Aug. 1992.
[15] B. Lakshmi, and A. S. Dhar, "FPGA Implementation of a High Speed VLSI Architecture for CORDIC," IEEE TENCON 2009.
[16] R. Kunemund, H. Soldner, S. Wohlleben, T. Noll, "CORDIC Processor with Carry-Save Architecture," Sixteenth European Solid- State Circuits Conference, 1990. ESSCIRC '90, pp.193-196, 19-21 Sept. 1990.
[17] K. S. Yeo and K. Roy, Low-Voltage, Low-Power VLSI Subsystem, Tata Mcgraw-Hill, New Delhi, 2009.
[18] E. O. Garcia, R. Cumplido and Miguel Arias, "Pipelined CORDIC Design on FPGA for a Digital Sine and Cosine Waves Generator," 2006 3rd International Conference on Electrical and Electronics Engineering(ICEEE),pp.104-107, Sept.2006.
[19] Chua-Chin Wang, Chia-Hao Hsu, Tuo-Yu Yao and Jian-Ming Huang, "A ROM-less DDFS Using A Nonlinear DAC With An Error Compensation Current Array," IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2008, pp.1632-1635, Nov. 30 2008-Dec. 3 2008.
[20] Chua-Chin Wang, Jian-Ming Huang, Y.-L. Tseng, Wun-Ji Lin and Ron Hu, "Phase-Adjustable Pipelining ROM-Less Direct Digital Frequency Synthesizer With a 41.66-MHz Output Frequency ," IEEE Transactions on Circuits and Systems—II: Vol.53(10), pp.1143-1147, Oct. 2006.
[21] C. H. Roth Jr and L. K. John, Principles of Digital Systems Design using VHDL, Cengage Learning, New Delhi, 2008