International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 6, Issue 11 (April 2013), PP. 08-14

FPGA implementation of universal modulator using CORDIC algorithm for communication application

G. HariKumar[1], M. MadhuBabu,M.Tech(Ph.D)[2],
[1](ECE Department, GPR College of Engineering,Kurnool, Indian).
[2](ECE Department,Assistant Professor of GPR College of Engineering, Kurnool, Indian.

Abstract:- The modern communication systems and software radio based applications demands fully digital receivers, consisting of only an antenna and a fully programmable circuit with digital modulators and demodulators. A basic communication system's transmitter modulates the amplitude, phase or frequency proportional to the signal being transmitted. An efficient solution (that doesn't require large tables/memory) for realizing universal modulator is CORDIC (CO-ordinate Rotation Digital Computer) algorithm. The CORDIC algorithm is used in the rotation mode, to convert the coordinates from polar mode to rectangular mode. The radius vector(r) and angle (θ) of a CORDIC processor can be programmed to generate the ASK, PSK and FSK signals.Modelsim simulator tool from mentor graphics will be used for functional simulation and verification of the modulator. The Xilinx synthesis Tools (XST) will be used to synthesize the complete modulator on Xilinx Spartan 3E family FPGA (XC3S500E). Xilinx placement & routing tools will be used for backed, design optimization and I/O routing.

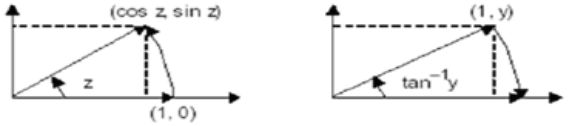
Keywords:- CORDIC, ASK, PSK, FSK, FPGA.

I. INTRODUCTION

Coordinate Rotation Digital Computer (CORDIC) algorithms provide an efficient approach for rotating vectors in a plane by simple shift-add operations. Besides offering a simple mechanism for computing the magnitude and phase-angle of an input vector, they are popularly used to perform complex multiplications and estimation of various functions. The conventional CORDIC algorithms are used to handle rotations only in the range of $[-99.7^0,99.7^0]$. Moreover, they are serial in nature and require a ROM to store the look-up table and hardware-expensive barrel shifters. Research on enhancements to the conventional CORDIC has proceeded primarily intwo directions. One of these is to obtain a high speed solution while the other is on careful handling of scale factors for high precision implementations. The authors in describe a virtually scaling-free CORDIC rotator which requires a constant scale-factor. A solution that scales the vector after every iteration, reduces the number of iterations and expands the range of rotation angles is presented in. A CORDIC architecture that suggests a circuit for scale factor correction is presented in some efforts have also been made to implement the CORDIC architectures on an FPGA.

The contributions of this paper are as follows: We present two highly area-efficient CORDIC algorithms with an angular resolution of. Convergence of both the algorithms is established theoretically. The algorithms cover the entire range of angles $[-180^0,180^0]$. Further, the FPGA implementations consume a substantially lower percentage of device components than other implementations.

II. CORDIC algorithm key ideas



Start at(1,0) Start at (1,y)
Rotate by z Rotate until y=0
Get cos z, sin z rotation amount is $\tan^{-1}y$

if we have a computationally efficient way of rotating a vector, we can evaluate cos,sin and $\tan^{-1}$ functions. Rotation by an arbitrary angle is difficult, so we perform psuedorotations use special angles to synthesize a desired angle z

$$z=\alpha^{(1)}+\alpha^{(2)}+\ldots\ldots\ldots+\alpha^{(m)}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = x\cos\varphi - y\sin\varphi$$

$$y' = y\cos\varphi + x\sin\varphi$$

$x' = \cos\varphi[x - y\tan\varphi]$

$y' = \cos\varphi[y + x\tan\varphi]$

$\tan\varphi = 2^{-i}$

$x_{i+1} = k_i[x_i - y_i.d_i.2^{-i}]$

$y_{i+1} = k_i[y_i + x_i.d_i.2^{-i}]$

$k_i = \cos(\tan^{-1}2^{-i}) = 1/\sqrt{1+2^{-2i}}$

$d_i = \pm 1$

$$x_{i+1} = x_i - y_i.d_i.2^{-i}$$

$$y_{i+1} = y_i + x_i.d_i.2^{-i}$$

$$z_{i+1} = z_i - d_i.tan^{-1}(2^{-i})$$

Where d_i=-1 if z_i<0,+1 otherwisedi= means direction ,ki=scaling factor

Removing the scale constant from the iterative equations yields a shift add algorithm for vector rotation. The product of the Ki's can be applied elsewhere in the system or treated as a part of the system processing gain. The product approaches 0.6073 as the number of iteration goes to infinity. Therefore the rotation algorithm has a gain An of approximately 1.647.The exact gain depends on the no of iterations and obeys the relation

$$A_n = \pi_n\sqrt{1+2^{-2i}}$$

The angle of a composite rotation is uniquely defined by the sequence of the directions of the elementary rotations. That sequence can be represented by a decision vector. The set of all possible decision vectors is an angular measurement system based on binary arctangents. Conversions between the angular systems and any other can be accomplished using a look up. A better conversion method uses an additional adder-subtract or that accumulate the elementary rotation angles at each iteration. The elementary angles can be expressed in any convenient angular unit. Those angular values are supplied by a small look up table or are hardwired depending on the application.

The angle accumulator adds a third difference equation to the CORDIC algorithm:

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

Obviously, in cases where the angle is useful in the arc tangent base, this extra element is not needed.

III. CORDIC TECHNIQUE

Given a complex value:$C=I_C+jQ_C$

We will create a rotated value:$C'=I_{C'}+jQc$

by multiplying by a rotation value:$R=Ir+jQr$

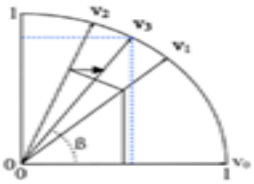


Fig.1 Rotation vector

How multiplier less:

1.Recall that when you multiply a pair of complex numbers, their phases (angles) add and their magnitudes multiply. Similarly, when you multiply one complex number by the conjugate of the other, the phase of the conjugated one is subtracted (though the magnitudes still multiply).

Therefore:

To add R's phase to C:

C' = C·R	Ic' = Ic·Ir-Qc·Qr
	Qc' = Qc·Ir+Ic·Qr

To subtract phase from C:

C' = C·R*	Ic' = Ic·Ir+Qc·Qr
	Qc' = Qc·Ir-Ic·Qr

2.To rotate by +90 degrees, multiply by R = 0 + j1. Similarly, to rotate by -90 degrees, multiply by R = 0 - j1. If you go through the Algebra above, the net effect is:

To add 90 degrees Multiply by R=0+j1

$I_C' = -Q_C$ $Q_C' = I_C$	(negate Q,then swap)

To Subtract 90 degrees multiply by R=0-j1

$I_C' = Q_C$ $Q_C' = -I_C$	(negate I, then swap)

3.To rotate by phases of less than 90 degrees, we will be multiplying by numbers of the form "R = 1 +/- jK". K will be decreasing powers of two, starting with 2^0 = 1.0. Therefore, K = 1.0, 0.5, 0.25, etc. (We use the symbol "L" to designate the power of two itself: 0, -1,-2,etc.) Since the phase of a complex number "I + jQ" isatan(Q/I), the phase of "1 + jK" is atan(K). Likewise, the phase of "1 - jK" = atan(-K) = -atan(K).To add phases we use "R = 1 + jK"; to subtract phases we use "R = 1 - jK". Since the real part of this, Ir, is equal to 1, we can simplify our table of equations to add and subtract phases for the special case of CORDIC multiplications to:

To add a phaseMultiply by R=1+Jk

Ic' = Ic-K·Qc =Ic(2^L)·Qc Qc'=Qc+K·Ic=Qc+(2^-L)·Ic

To add a phaseMultiply by R=1+Jk

Ic'=Ic+K·Qc=Ic + (2^-L)·Qc Qc'=Qc-K·Ic = Qc- (2^-L)·Ic

Table 1 Effect by multiplying the complex number

L	K = 2^-L	R = 1+jK	Phaseof R indegrees = atan(K)	Magnitude of R	CORDIC Gain
0	1.0	1+j1.0	45.00000	1.41421356	1.414213562
1	0.5	1+j0.5	26.56505	1.11803399	1.581138830
2	0.25	1+j0.25	14.03624	1.03077641	1.629800601
3	0.125	1+j0.125	7.12502	1.00778222	1.642484066
4	0.0625	1+j0.0625	3.57633	1.00195122	1.645688916
5	0.03125	1+j0.031250	1.78991	1.00048816	1.646492279
…	…	…	…	…	…

4.Each rotation has a magnitude greater than 1.0. That isn't desirable, but it's the price we pay for using rotations of the form "1 + jK". The "CORDIC Gain" column in the table is simply a "cumulative magnitude" calculated by multiplying the current magnitude by the previous magnitude. Notice that it converges to about 1.647; however, the actualCORDIC Gain depends on how many iterations we do. (It doesn't depend on whether we add or subtract phases, because the magnitudes multiply either way).

(i) CORDIC BASIC ISSUES

- Since we're using powers of two for the K values, we can just shift and add our binary numbers. That's why the CORDIC algorithm doesn't need any multiplies.
- The sum of the phases in the table up to L = 3 exceeds 92 degrees, so we can rotate a complex number by +/- 90 degrees as long as we do four or more "R = 1 +/- jK" rotations. Put that together with the ability to rotate +/-90 degrees using "R = 0 +/- j1", and you can rotate a full +/-180 degrees.
- You can see that starting with a phase of 45 degrees, the phase of each successive R multiplier is a little over half of the phase of the previous R. That's the key to understanding CORDIC: we will be doing a "binary search" on phase by adding or subtracting successively smaller phases to reach some "target" phase.
- Each rotation has a magnitude greater than 1.0. That isn't desirable, but it's the price we pay for using rotations of the form "1 + jK". The "CORDIC Gain" column in the table is simply a "cumulative magnitude" calculated by multiplying the current magnitude by the previous magnitude. Notice that it converges to about 1.647; however, the actual CORDIC Gain depends on how many iterations we do. (It doesn't depend on whether we add or subtract phases, because the magnitudes multiply either way.)

Modes of operation:
Basic mode: I and Q carrier signals for a chosen freq value
Modulator mode: producing ASK, FSK and PSK signals.
11-bit Control word format

M	M	F	F	F	F	F	F	F	K	K

MM --- Mode of operation 00 – Basic mode 01 – Binary Modulator mode 10 – Mary modulator mode FFFFFFF – 7-bit Freq word KK (in Binary modulation mode) 00 – Amplitude shift keying 01 –Phase shift keying 10 – Freq shift keying KK(in M-ary modulation mode) 00 – 4 level QAM 01 – QPSK 10– OQPSK 11-8psk.

IV. BLOCK DIAGRAM OF CORDIC BASED UNIVERSAL MODULATOR REALIZATION

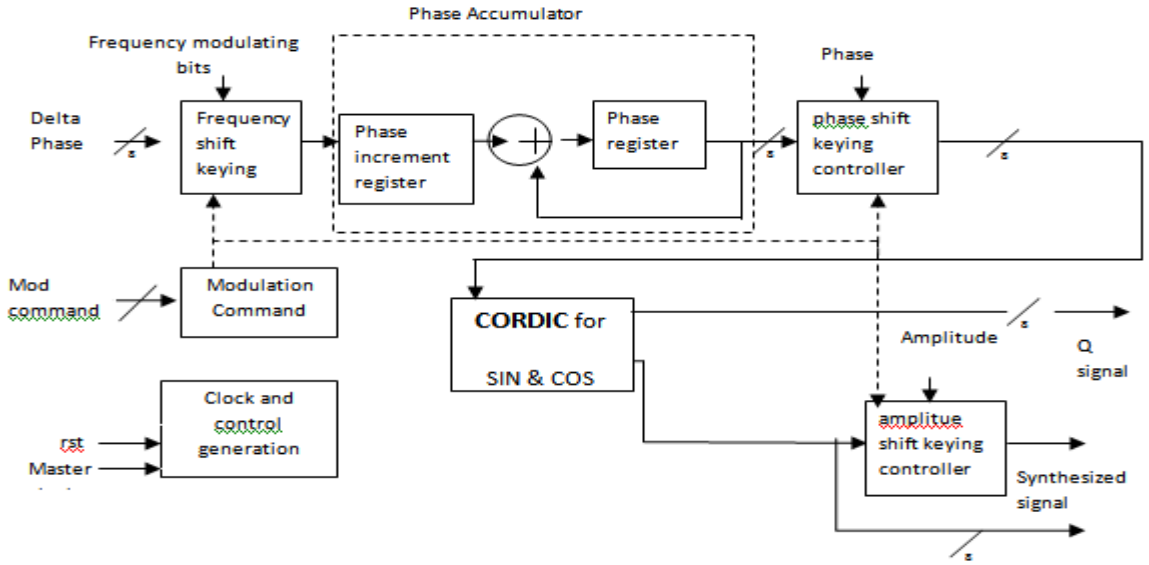


Fig.2 block diagram of CORDIC

(I).Basic Blocks Of Universal Modulator

All the blocks are connected with common clock and reset signals. The delta phase value decides the phase increment for each clock pulse. Hence decides the resulting signal frequency. The Frequency modulating instantaneous value is added to the delta phase value which causes instant-aneous change in frequency. Due to the digital nature of the modulator only at each clock tick the modulating signal value shall affect the resulting frequency.The outputs of the Look Up Tables are given to the input lines a 4 to 1 Multiplexer.This multiplexer connects one of the inputs to the output depending on the select lines. The output of Multiplexer consists the 8 amplitude bits which is the final output in case required modulation schemes are FM or PM. In case of Amplitude modulation, the output of multiplexer is multiplied with instaneous modulating signals. CORDIC engine isused for phase to amplitude conversion required for the generation of cosine and sin functions.
In three modulation schemes if modulating signal is analog in nature then anappropriate Analog toDigital converter is required to convert into 8 bitdigital output. From the figure the basic blocks in DDFS can be identified as PIPO registers, adders, look Up Tables, CORDIC engine and other combinational circuits.

(ii) DDFS Basic Principle:

Direct Digital Frequency Synthesizer is a technique to produce desired output wave-forms with full digital control. Direct digital synthesis (DDS) is becoming increasingly popular as a technique for fre-quency synthesis, especially if high freque-ncy resolution and fast switching between frequencies over a large bandwidth are required.
The direct digital frequency synthesizer is shown in a simplified form in figure 4. The DDFS has the following basic blocks:
(1) Frequency register
(2) adder and phase register
(3) phase to amplitude converter (conventionally a sine ROM)
(4) digital to analog converter and low pass filter

The phase value is generated using the modulo 2^j overflowing property of a j-bit phase accumulator. The rate of the overflows is the output frequency

$$f_{out} = \frac{\Delta P\, f_{clk}}{2^j} \;\forall\; f_{clk} \leq \frac{f_{clk}}{2} \quad \text{-- (1.1)}$$

Where ΔP is the phase increment word, j is the number of phase accumulator bits, $f_{clk}$is the clock frequency and $f_{out}$is the output frequency. The constraint for maximum value of f_{out} in the above equation comes from the sampling theorem.

The phase increment word in (1.1) is an integer, therefore the frequency resolution is found by setting $\Delta P = 1$.

$$\Delta f = (f_{clk})/2^j \quad \text{--- (1.2)}$$

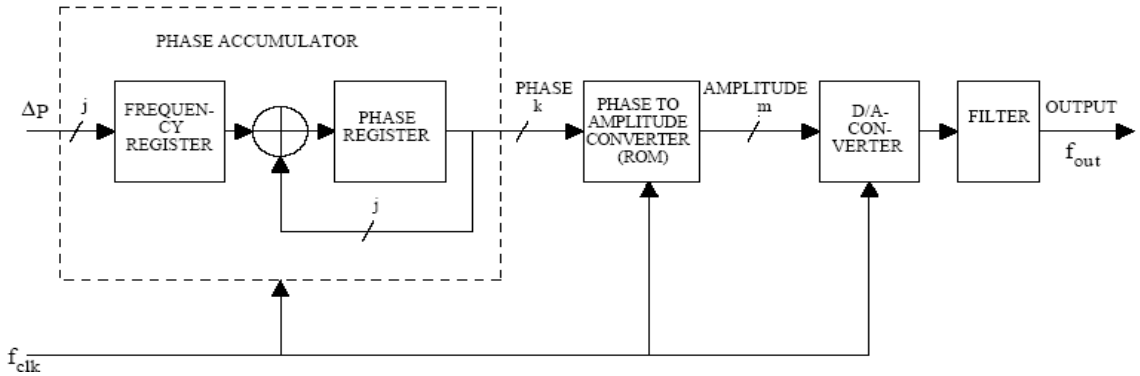


Fig.3: block diagram of DDFS

In the ideal case with no phase and amplitude quantization, the output sequence of the table is given by

$$\sin(2\pi\frac{P(n)}{2^j}) \text{--- --(1.3)}$$

Where $P(n)$ is a (the j-bit) phase register value (at the nth clock period). The numerical period of the phase accumulator output sequence is defined as the minimum value of Pe for which P(n)=P(n+Pe) for all n. The numerical period of the phase accumulator output sequence is

$$p_e = \frac{2^j}{GCD(\Delta p, 2^j)} \text{----- (1.4)}$$

Table 2

For an accumulator of 3 bits (j=3) controlled with an input of ΔP = 3 and ΔP =2

Accumulator output ΔP=3 and j=3	Carry output	Accumulator output Δp=2 and j=3	Carry output
000 (0)	1 Cycle begins	000 (0)	1 Cycle begins
011 (3)	0	010(2)	0
110(6)	0	100(4)	0
001(1)	1	110(6)	0
100(4)	0	000 (0)	1

(iii). DFS Architecture for Modulation capability:

It is simple to add modulation capabilities to the DDS, because the DDS is a digital signal processing device. In the DDS it is possible to modulate numerically all three wave form parameters.

$$S(n)=A(n)\sin(2\pi(\Delta P(n)+P(n))) \quad \text{---(1.5)}$$

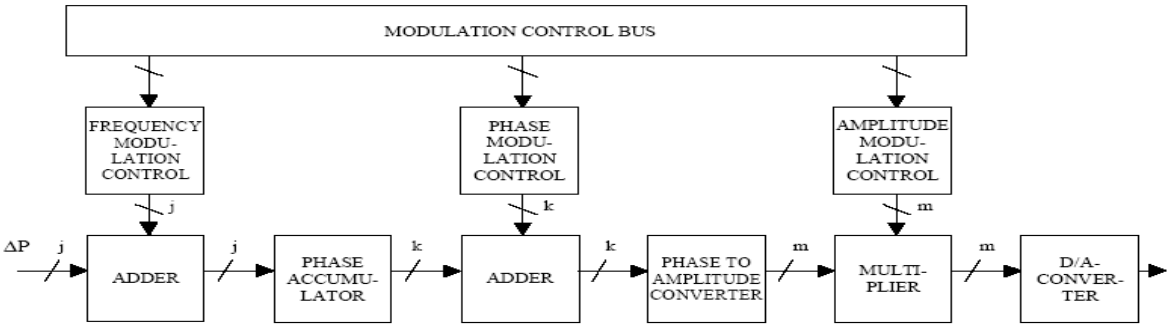


Fig.4 DDS architecture with modulation capabilities

V. MODULATION COMMAND PROCESSOR SIMULATION RESULTS

The Modulation command processor implemented are for carrying out digital versions of frequency modulation, phase modulation and amplitude modulation. For all these three modulations the digital 8 bit modulating signal is expected. In case if the modulating signal is analog that needs to be converted in to 8 bit digital by appropriate Analog to Digital Converter.

The frequency modulation is carried out by frequency modulation controller which is simply an adder adding the phase increment value input for DDFS and instantaneous modulating signal value. Since the phase increment value or delta phase value is proportional to instantaneous frequency of output signal of DDFS, controlling that parameter leads to Frequency modulation. The phase modulation controller adds the output of phase accumulator to the instantaneous value of phase modulating signal. The resulting sum is fed tothe LUTs to produce amplitude bits corresponding to the input phase bits. The amplitude modulator planned in this project is analogous to product modulator. The amplitudebits at the output of multiplexer are multiplied with instantaneous amplitude of modulating signal.

To verify the modulation effects from these modulation controllers, these blocks should work in complete DDFS module; hence the resulting outputs for three types of modulations are presented

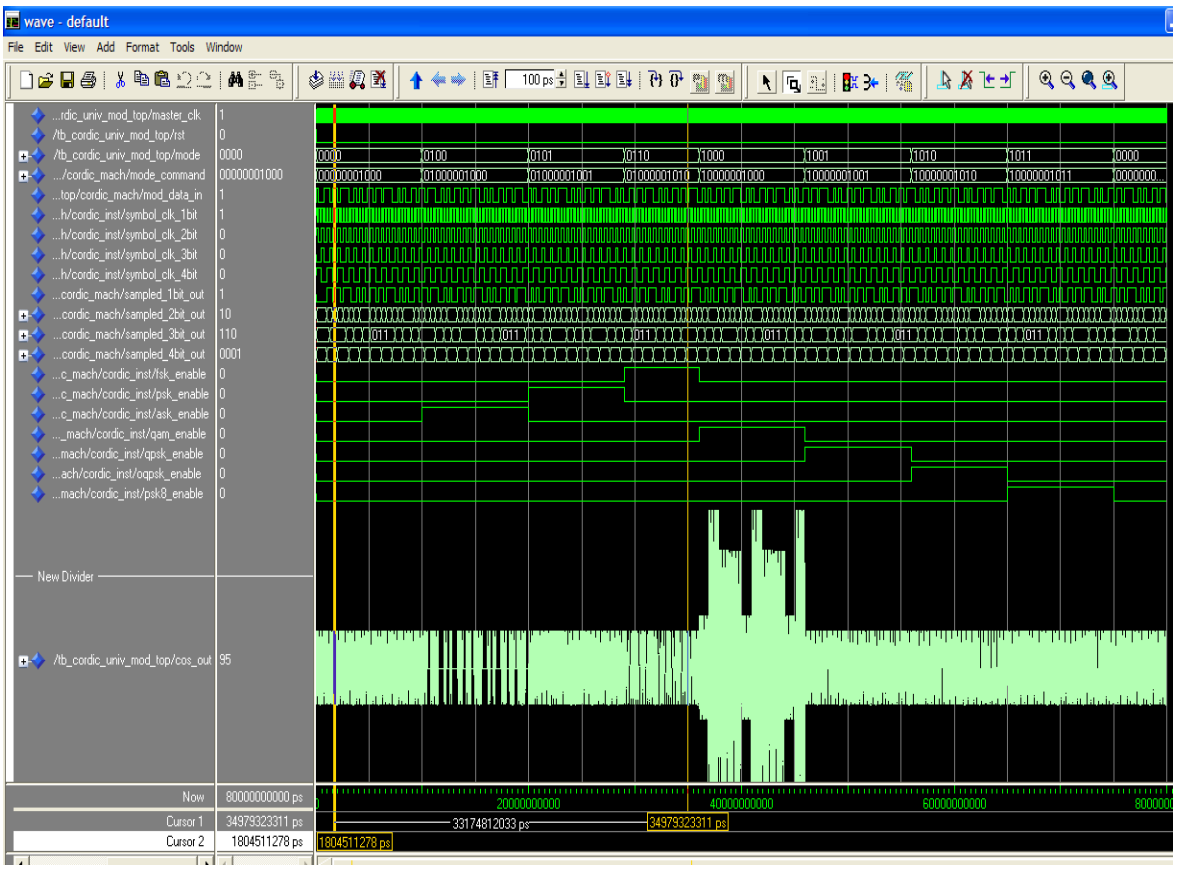


Fig.5 Modulation command processor simulation results

VI. APPLICATIONS

- Universal modulator is highly preferred option in instrumentation for signal generation and frequency sweep etc.
- Universal modulator can be efficiently used in several digital modulation schemes suchas FSK, PSK, ASK, QPSK, OPQPSK, PI/QPSK, QAM and 8-bitPSK.
- Universal modulator has been used in universal HAM radio/generator. Universal modulator is capable of well controlled, rapid changes in frequency.

VII. ADVANTAGE AND DISADVANTAGES

- High frequency resolution and accuracy.
- Fast switching between frequencies over a large bandwidth.
- Compared to LUT method, it takes less area.
- FPGA hardware can give high speed solution in comparison with software approach but not in comparison with ASIC.

VIII. TOOLS AND HARD WARE

- Simulation software -Modelsim Xilinx Edition (MXE)
- Synthesis, P&R - Xilinx ISE
- On chip verification - Xilinx Chip scope
- Hardware– Xilinx Spartan 3 Family FPGA board

IX. FPGA DESIGN AND PROGRAMMING

To define the behavior of the FPGA, the user provides a hardware description language (HDL) or a schematic design. The HDL form is more suited to work with large structures because it's possible to just specify them numerically rather than having to draw every piece by hand. However, schematic entry can allow for easier visualization of a design.Once the design and validation process is complete, the binary file generated (also using the FPGA company's proprietary software) is used to (re)configure the FPGA.

X. CONCLUSION

CORDIC algorithms are an efficient method it is applicable to the entire range of angles are presented in this paper. Convergence proofs for the new algorithms are also presented. Both the algorithms consume a substantially lower percentage of (FPGA) device components incomparison to prior algorithms. Further, the FPGA implementations exhibit excellent performance of the proposed schemes in terms of the slice-delay products.

- In some cases, CORDIC evaluates rotational functions more efficiently than MAC units.
- CORDIC saves more hardware cost.
- By the regularity, the CORDIC based architecture is very suitable for implementation with pipelined VLSI array processors.
- The utility of the CORDIC based architecture lies in its generality and flexibility.

REFERENCES

[1]. J.VolderThe CORDIC trigonometric computing technique, IRE Transactions on Electronic computers, Vol.EC 8, 2010, Pp.330-334.

[2]. R.Andraka, A survey of CORDIC algorithms for FPGA based computers, Proceedings of ACM/SIGDA sixth International Symposium on field Programmable Gate Arrays, 1998,pp.191-200

[3]. J.S.Walther, A unified algorithm for elementary functions, proceedings of 38th spring joint computer conference, 2000, pp.379-385.

[4]. Efficient CORDIC Algorithms and Architectures for Low Area and High Throughput Implementation IEEE paper 2009.

[5]. A VLSI implementation of Logarithmic and Exponential Functions Using a Novel Parboil Synthesis Methodology Compared to the CORDIC Algorithms IEEE paper 2011.

[6]. T.Juang, S.Hsiao, and M.Tsai,"Parallel CORDIC rotation algorithm,"IEEETrans.CircuitsSyst.I:Reg.papers, Vol.51, no.8, pp.1515-1524, Aug.2004