International Journal of Engineering (IJE)
Singaporean Journal of Scientific Research(SJSR)
Vol 6.No.3 2014 Pp.143-156
Available at:www.iaaet.org/sjsr
Paper Received:12-03-2014
Paper Accepted:29-03-2014
Paper Reviewed by:1 Prof .Dr.Ponnambalam 2. Chai ChengYue
Editor:Dr. BinodKumar

SCALE-FREE HYPERBOLIC CORDIC PROCESSOR AND ITS APPLICATION TO WAVEFORM GENERATION

S.Kundavaipriya, N.Paruvatham
Department of ECE (VLSI Design), Moogambikai College of Engineering
Kalamavur, Keeranur, Pudhukottai.

ABSTRACT

This paper presents a novel completely scaling-free CORDIC algorithm in rotation mode for hyperbolic trajectory. We use most-significant-1 bit detection technique for micro-rotation sequence generation to reduce the number of iterations. By storing the sinh/cosh hyperbolic values at octant boundaries in a ROM, we can extend the range of convergence to the entire coordinate space. Based on this, we propose a pipeline hyperbolic CORDIC processor to implement a direct digital synthesizer (DDS). The DDS is further used to derive an efficient arbitrary waveform generator (AWG), where a pseudo-random number generator modulates the linear increments of phase to produce random phase-modulated waveform. The proposed waveform generator requires only one DDS for generating variety of modulated waveforms, while existing designs require separate DDS units for different type of waveforms, and multiple DDS units are required to generate composite waveforms. Therefore, area complexity of existing designs gets multiplied with the number of different types waveforms they generate, while in case of proposed design that remains unchanged. The proposed AWG when mapped on Xilinx Spartan 2E device, consumes 1076 slices and 2016 4-input LUTs. The proposed AWG involves significantly less area and lower latency, with nearly the same throughput compared to the existing CORDIC-based designs.

Index Terms—Direct digital synthesis, scale-free CORDIC, sigmoid functions, waveform generator.

1. INTRODUCTION

SIGNAL generators [1] are an important part in designing, testing and troubleshooting of various electronic systems. Arbitrary and non-linear waveforms of various forms are required in research and development to provide the necessary stimuli to the device under test (DUT). Normally, arbitrary waveform generators (AWGs) are combined with conventional function generators which include various predefined functions like sine, ramp, triangle, compound sinusoidal waveforms, exponentials, etc. The waveforms related to sinusoids are generated using circular trigonometry, whereas, the exponential or hyperbolic waveforms are generated using hyperbolic Manuscript received December 23, 2011; revised March 11, 2012; accepted April 04, 2012. Date of publication September 28, 2012; date of current version January 24, 2013. This paper was recommended Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org. Digital Object Identifier 10.1109/TCSI.2012.2215778 trigonometry. For proper testing it is necessary that the AWGs produce accurate and predictable waveforms. There are several methods of arbitrary waveform generation. The look-up-table

(LUT) approach is the simplest of those, but its area complexity is huge owing to large memory requirements. Typically, with a modest frequency resolution of 200 Hz, the AWG requires more than words in LUT for a maximum frequency of 100 MHz [2]. Moreover, the memory requirement increases linearly with the improvement of frequency resolution or the increase of maximum frequency. To overcome this disadvantage of LUT-based AWG, direct digital synthesizers (DDS) have gained preference for realizing high-frequency signal-generators with good frequency resolution. To generate composite waveforms, the outputs of various DDS units running at different frequencies are required to be added together [3]. Therefore, to increase the arbitrariness of the generated amplitudes, more number of DDS are required to be used in the AWG. But, the hardware cost of AWG increases linearly with the number of DDS units used in it. In [4], a technique based on Chebyshev polynomials is used for generating composite and arbitrary waveforms. It approximates a given waveform pattern using Chebyshev polynomials, and realizes a circuit for generating it repetitively using multipliers and adders. This paper considers realizing a waveform generator using CORDIC based DDS architecture. Instead of using conventional DDS, where phase is mapped to sinusoidal amplitude; in this paper, we propose to map the phase to cosine hyperbolic amplitude. As a result, we can obtain large variations in amplitude without amplitude scaling. In the proposed design, the arbitrary amplitude variations occur due to random phase modulation of the DDS. Unlike the existing techniques, the proposed technique requires only one DDS for generating various modulated/arbitrary waveforms. Hyperbolic modulated waveforms have a unique property of Doppler-shift invariance and therefore, have gained importance in the field of radar/sonar communications [5], [6]. The proposed AWG can be used for testing of these circuits. The waveforms generated by the proposed AWG have large number of frequency components, so it can serve as stimuli to test frequency selectivity of devices.

Apart from this, it can also also be used as a digital pattern generator. CORDIC architectures have been successfully employed for waveform generation [7], [8], implementation of digital filters [9], transform computation [10], [11], matrix calculations [12] etc. In spite of its simplicity and low computational complexity, CORDIC algorithm suffers from major bottlenecks like either high latency or large overheads of scale-factor compensation, when an optimized set of micro-rotations are used to reduce the latency. Parallel CORDIC architectures have been suggested in [13] and [14] to reduce the latency but at the cost of additional hardware and time to implement the scale-factor compensation. The redundant iterations are eliminated by greedy search in [15]–[17], but the hardware savings achieved by this approach are counter-balanced by variable scale-factor compensation circuits. Various scale-factor compensating techniques have been suggested in the literature [18]–[20], but these techniques either lead to large area overheads or otherwise affect throughput or latency. The Taylor series expansion offers a low complexity solution for the design of scale-free CORDIC. Scaling-Free CORDIC [21]–[23] indemnify the scale-factor limitations to certain extent. Various optimization efforts in the above CORDIC algorithms are targeted for circular CORDIC, while hyperbolic CORDIC still needs to be explored for improvements.

The number of efficient CORDIC designs for hyperbolic trajectory is far less as compared to circular trajectory regardless, inspite of its wide scope in artificial neural networks [24]–[26], adaptive filtering [27] and for computing logarithm and exponential functions [28]. In [29], the authors improve the range of convergence of conventional CORDIC algorithm in hyperbolic trajectory by using additional iterations which allow negative iteration indices as well. Though it increases the RoC of the hyperbolic CORDIC algorithm, it significantly adds to the latency of the processor.

Themain contributions of this paper are: (i) a novel scale-free design of hyperbolic CORDIC1 using optimized micro-rotation sequence with improved RoC, (ii) a DDS using the proposed scale-free hyperbolic CORDIC processor for generating hyperbolic/ exponential waveforms, and (iii) a novel scheme of random phase modulation for generation of arbitrary waveforms. The rest of the paper is organized as follows. An overview of CORDIC algorithm is presented in Section. II. The proposed technique for the design of scale-free hyperbolic CORDIC is discussed in Section III. Section IV deals with the optimization of the micro-rotation sequence, while Section V deals with convergence- range of the proposed CORDIC algorithm. Section VI details the error analysis of the proposed methodology. The design of DDS and the waveform generator based on the proposed scale-free hyperbolic CORDIC are described in Section VII. FPGA implementation and complexity issues are discussed in Section VIII with a brief conclusion in Section IX.

2. CORDIC ALGORITHM

The CORDIC algorithm operates either in rotation or vectoring mode, following linear, circular or hyperbolic trajectories. The circular CORDIC algorithm is used for computation of sin/cos, vector rotations etc., while hyperbolic CORDIC is used for calculating exponents, sinh/cosh etc. In this paper, we focus on rotation mode of CORDIC operation using hyperbolic trajectory. 1The proposed hyperbolic CORDIC works in rotation mode only. It does not support vectoring mode of CORDIC.

A. Unified CORDIC Algorithm

The unified CORDIC algorithm [30] is an extension to the basic CORDIC algorithm of Volder [31]. The generalized principle was proposed byWalther to include, hyperbolic and linear trajectory along with the original circular trajectory of operation. A variable for defining the trajectory was introduced to modify the basic CORDIC rotation matrix and elementary angle as:

$$\mathbf{R_p} = K_i \cdot \begin{bmatrix} 1 & -m \cdot \mu_i \cdot 2^{-i} \\ \mu_i \cdot 2^{-i} & 1 \end{bmatrix},$$

$$K_i = \frac{1}{\sqrt{1 + m \cdot 2^{-2i}}} \tag{1a}$$

$$\theta = \sum_{i=0}^{b} \mu_i \cdot \alpha_i, \quad \alpha_i = \frac{1}{\sqrt{m}} \cdot \tan^{-1}\left(\sqrt{m} \cdot 2^{-i}\right)$$

$$\text{where } m = \begin{cases} -1 & \text{for hyperbolic} \\ 0 & \text{for linear} \\ 1 & \text{for circular} \end{cases} \tag{1b}$$

The equations for hyperbolic trajectory are thus formulated in unified CORDIC algorithm as:

$$\mathbf{R_p} = K_i \cdot \begin{bmatrix} 1 & \mu_i \cdot 2^{-i} \\ \mu_i \cdot 2^{-i} & 1 \end{bmatrix} \text{ and } \alpha_i = \tanh^{-1}(2^{-i}) \tag{2}$$

To guarantee convergence in hyperbolic mode, the iterations , 13, 40 need to be executed twice. Consequently, the scale factor converges to a constant . The CORDIC algorithm in hyperbolic rotation mode can support a RoC of .

B. Review of Scaling-Free CORDIC

The Scaling-Free CORDIC [21] is a milestone in development of optimized implementation of CORDIC algorithm. It lays the foundation for using the Taylor series approximation in CORDIC rotation matrix for deriving the scale-free equations. It employs second order Taylor series approximation. Though originally designed for circular CORDIC applications, it needs a brief mention as, it discusses the implications of Taylor series approximation on RoC and the permissible highest elementary angle that could be used for realizing CORDIC rotations. The rotation matrix for Scaling-Free CORDIC is given as: (3)

This approximation imposes a restriction on the basic-shift2 . For 16-bit applications, the basic-shift is, which reduces the ROC to 7.16 . This was a major drawback, which limits the applicability of this algorithm. Moreover, this algorithm focuses only on circular rotation mode, and cannot be directly extended to hyperbolic CORDIC. The second order of approximation of Taylor series expansion of hyperbolic functions would lead to a very low range of convergence of 7.16 . Due to absence of any kind of symmetry in hyperbolic functions one cannot expand the RoC, as is done in [21] to expand the RoC of circular trigonometric functions. 2Theminimum possible permissible shifts in the CORDIC iteration have been termed as basic shift, which is equal to the number of right shifts in the first CORDIC iteration.

3. PROPOSED SCALING-FREE HYPERBOLIC CORDIC

The main steps in the design of the proposed scaling-free hyperbolic CORDIC with enhanced range of convergence (RoC) are: (i) Modification of the unified CORDIC coordinate equations using Taylor series approximation of hyperbolic terms to derive scale-free CORDIC equations3; (ii) Determination of the sequence of micro-rotations based on the most-significant-1 location in the radix-2 representation of the rotation angle, and also restricting the micro-rotations in single direction; and (iii) Formulation of necessary mathematical relations to increase the RoC of the unified CORDIC algorithm.

In this section, we elaborate the proposed scale-free coordinate calculation unit, while the micro-rotation sequence generation is detailed in the next section. To derive a scale-free rotation matrix for hyperbolic coordinate computations, we use the hyperbolic rotation matrix in the form:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \mathbf{R_p} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \qquad \mathbf{R_p} = \begin{bmatrix} \cosh \alpha_i & \sinh \alpha_i \\ \sinh \alpha_i & \cosh \alpha_i \end{bmatrix} \tag{4}$$

Using Taylor series expansion of sinh and cosh terms in the rotation matrix (4) can get rid of the scale-factor of the rotation matrix (2) for the unified CORDIC algorithm.

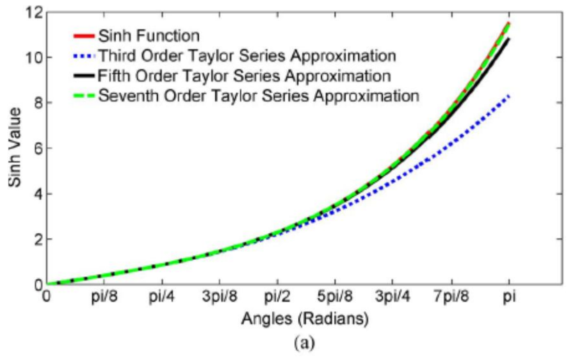
A. Taylor Series Approximation

The Taylor series expansions of the hyperbolic functions are given by:

$$\sinh \alpha = \alpha + \frac{\alpha^3}{3!} + \frac{\alpha^5}{5!} + \cdots$$
$$\cosh \alpha = 1 + \frac{\alpha^2}{2!} + \frac{\alpha^4}{4!} + \cdots \tag{5}$$

For hardware implementation the above expansions need to be approximated, with acceptable compromise in accuracy. Also, the Taylor series approximations employed in the CORDIC rotation matrix affect the RoC, as it restricts the basic-shift (Appendix A (28b)). However, we find that the increase in the approximation order increases the RoC by decreasing the basic-shift. But at the same time, it adds to the hardware complexity as additional terms are included in the implementation. Therefore, depending on the accuracy requirements and desired RoC, we need to select an appropriate order of approximation.

To decide the approximation order of the Taylor series to be used in CORDIC rotation matrix, we analyze various orders of approximation depending on the accuracy associated with each order. In Fig. 1(a) and Fig. 1(b) we have plotted sinh and cosh values for different orders of approximation of Taylor series respectively, for angles lying in the range . From Fig. 1 we see that, the original and approximated values are indistinguishable for sinh and cosh function up to , which covers the range of CORDIC elementary angles of . Themaximum percentage error in sinh and cosh values for third order of approximation is 0.00322% and 0.0158%, respectively. Therefore, 3The order of approximation of Taylor series plays a significant role in deciding the RoC of the CORDIC processor.



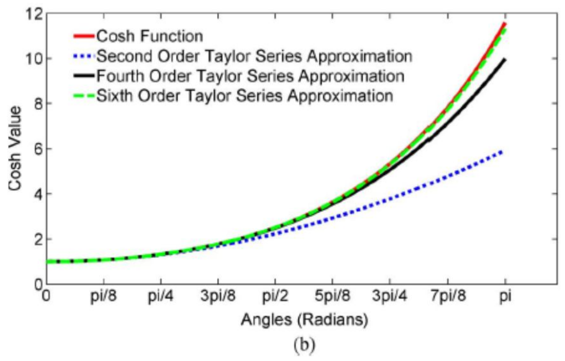


Figure. 1. Comparison of hyperbolic functions and Taylor series approximation.

(a) Comparison of sinh function and its Taylor series approximation;
(b) comparison of cosh function and its Taylor series approximation. we can safely select third order of approximation for Taylor series expansion of sinh and cosh terms to realize CORDIC rotations in hyperbolic rotation mode applicable for most practical applications.

B. Realizing CORDIC Iterations Using Taylor Series

To analyze the errors for CORDIC coordinate equations, we simulate the CORDIC rotator for various approximation orders of Taylor series. We realize the CORDIC processor for the following rotation matrices (derived by using various orders of approximation of Taylor series (5) in (4)):

$$R_{p1} = \begin{bmatrix} 1+\frac{\alpha_i^2}{2!} & \alpha_i+\frac{\alpha_i^3}{3!} \\ \alpha_i+\frac{\alpha_i^3}{3!} & 1+\frac{\alpha_i^2}{2!} \end{bmatrix} \quad (6a)$$

$$R_{p2} = \begin{bmatrix} 1+\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} & \alpha_i+\frac{\alpha_i^3}{3!} \\ \alpha_i+\frac{\alpha_i^3}{3!} & 1+\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} \end{bmatrix} \quad (6b)$$

$$R_{p3} = \begin{bmatrix} 1+\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} & \alpha_i+\frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!} \\ \alpha_i+\frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!} & 1+\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} \end{bmatrix} \quad (6c)$$

$$R_{p4} = \begin{bmatrix} 1+\frac{\alpha_i^2}{2!} & \alpha_i+\frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!} \\ \alpha_i+\frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!} & 1+\frac{\alpha_i^2}{2!} \end{bmatrix} \quad (6d)$$

Other higher order approximations are not considered as they only add to the hardware complexity with no benefit in terms of desired accuracy. Fig. 2 plots the error in the vector end-point for the rotation matrices in (6), with respect to the MATLAB inbuilt functions. For the vector.

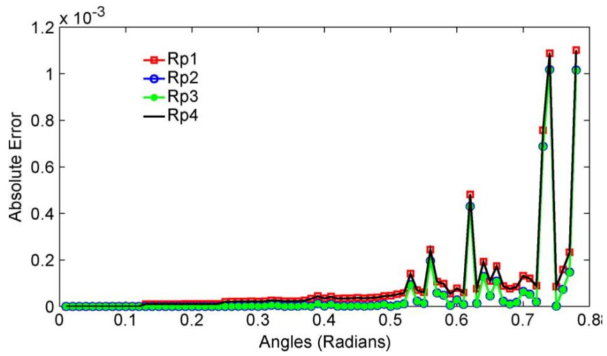


Figure. 2. Absolute error in vector end-point using (6) to determine order of approximation. the end-point is defined in (7).

The mean square error in the vector end-point, over the range of rotation angles , for the four rotation matrices (6) is , respectively. The error being small enough, we can say that the accuracy benefits are insignificant even if we increase the order of approximation. The third order of approximation thus satisfies our accuracy requirement with minimum hardware complexity. Therefore, we choose the third order of approximation to design our CORDIC processor and take the rotation matrix given by (6a).

$$V_{end-point} = \sqrt{x^2 + y^2} = \sqrt{\cosh^2\theta + \sinh^2\theta} \qquad (7)$$

C. Design of Coordinate Calculation Unit

The coefficients of Taylor series in (6a) do not allow shift-add implementation of rotationmatrix which is a mandatory requirement in CORDIC coordinate computations. In order facilitate shift-add implementation in the proposed rotation matrix (8), we approximate (3!) to . Therefore, the rotation matrix is modified to:

$$\mathbf{R_{p1}} = \begin{bmatrix} 1 + 2^{-1}\cdot\alpha_i^2 & \alpha_i + 2^{-2}\cdot\alpha_i^3 \\ \alpha_i + 2^{-2}\cdot\alpha_i^3 & 1 + 2^{-1}\cdot\alpha_i^2 \end{bmatrix} \qquad (8)$$

Assuming $\alpha_i = 2^{-s_i}$, where s_i is the shift for i^{th} iteration:

$$\mathbf{R_{p1}} = \begin{bmatrix} 1 + 2^{-(2s_i+1)} & 2^{-s_i} + 2^{-(3s_i+2)} \\ 2^{-s_i} + 2^{-(3s_i+2)} & 1 + 2^{-(2s_i+1)} \end{bmatrix} \qquad (9)$$

As the multiplications by power of 2 can be realized using left/right shift operations, the rotation matrix given by (9) can now be implemented using a shift-add network. The mean square error in the vector end-point using (9), over the range of convergence is . By the proposed approximation of rotation matrix, though the magnitude of the error increases slightly, the order remains the same.

IV. DECOMPOSITION OF ANGLE OF ROTATION INTO MICRO-ROTATIONS

In conventional CORDIC, the decomposition of angle of rotation is based on following conventions: (i) the elementary angles are pre-defined and stored in a ROM, (ii) the micro-rotation corresponding to all the elementary angles are performed, either

Table I Bit Representation of Elementary Angles And Corresponding Shifts

Shifts (s_i)	Elementary Angle (α_i) Decimal	16-bit Hexadecimal
2	0.25	4000H
3	0.125	2000H
4	0.0625	1000H
5	0.0312	0800H

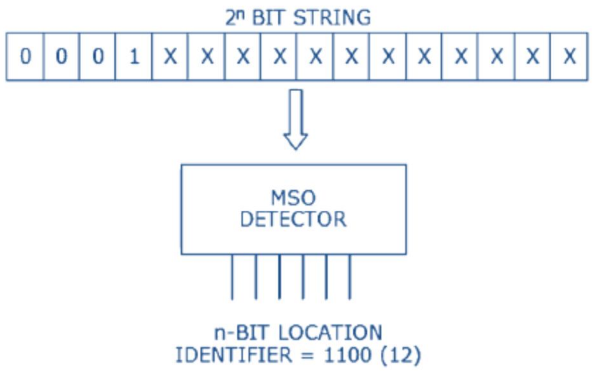

2ⁿ BIT STRING
0 0 0 1 X X X X X X X X X X X X
MSO DETECTOR
n-BIT LOCATION IDENTIFIER = 1100 (12)

Figure. 3. Functionality of MSO-LI with ‘x’ represent don’t care conditions.

clockwise or anti-clockwise, and (iii) each elementary angle is used only once. In the proposed scheme, we deviate from all these conventions. Themicro-rotations are confined to single direction with, multiple iterations corresponding to the basic-shift and non repetitive iterations for other shifts (Appendix B).

A. Redefinition of the Elementary Angles

Equation (2) defines the elementary angles used in the unified rotation-mode CORDIC algorithm. In order to simplify the design and eliminate the ROM for storing elementary angles, we redefine the elementary angles as:

$$\alpha_i = 2^{-s_i} \qquad (10)$$

where is the number of shifts for iteration In Table I we list the redefined elementary angles in decimal and 16-bit hexadecimal representation and the corresponding shifts.

B. Description of Most-Significant-One (MSO) Location

The most-significant-one (MSO) location refers to the bitposition of the leading-one in an input string of bits starting from most-significant-bit (MSB). The MSO Location Identifier (MSO-LI) generates an -bit output for a -bit input string, Fig. 3 shows the functionality of MSO-LI. From Table I, we observe the MSO-LI can be used to determine the number of shifts corresponding to the elementary angle using the formula:

$$s_i = N - MSO_{Location} \tag{11}$$

where is the word-length.

C. Determination of Highest Elementary Angle

The largest elementary angle depends on the order of approximation of Taylor series used to design the coordinate calculation unit. Using the formulas in Appendix A, the basic-shift and the

TABLE II
ERROR COMPARISON FOR VARYING n_2 ITERATIONS

n_2	% Error					
	$s_{basic} = 2$			$s_{basic} = 3$		
	Vector End-Point	Cosh	Sinh	Vector End-Point	Cosh	Sinh
4	0.0057	0.3009	0.5534	0.0034	0.1631	0.3355
5	0.0034	0.1080	0.5144	0.0007	0.0447	0.1256
6	0.0035	0.1659	0.5144	0.0007	0.0304	0.1256

largest elementary angle for the third order of approximation are
found to be:

$$s_{basic} = \left\lfloor \frac{b - \log_2(4!)}{4} \right\rfloor$$
$$\alpha_{max} = 2^{-s_{basic}} \tag{12}$$

where is the word-length
For 16-bit word-length, . Depending upon the desired accuracy, one can either select or
.

D. Determination of Micro-Rotation Sequence

We express any rotation angle as:

$$\theta = n_1 \cdot \alpha_{max} + \sum^{n_2 \text{ iterations}} \alpha_{s_i} \tag{13}$$

where and Note that the total number of iterations ‘ ’ is a constant. Depending on the angle of rotation we may or may not require iterations corresponding to ; for ‘ ’ to be a multiple of and for . We propose to identify the micro-rotations based on the binary representation of the rotation angles. To restrict the complexity of this block, the maximum angle of rotation handled by this block is set to be . Though the angle of rotation handled by micro-rotation sequence generation lies in the range , the RoC can be extended to any desired range as detailed in
the next section. We use 16-bit fixed point binary representation, where angle is represented as 1100_1001_0000_1111, while those of the elementary angles are tabulated in Table I.We design the micro-rotation sequence generator keeping in mind: 1) For : The highest elementary
angle would be used for the CORDIC iteration. *Example—I*: Let the rotation angle be radians be represented in 16-bit fixed point format as 1011_0010_1011_1000. The is 15. As which is , the elementary angle used will correspond to . Also from Table I, it can be observed that the elementary angle required to realize this rotation angle is 0.25 radians with shift-index . 2) For : The elementary angle would be used for the CORDIC iteration, where is given by (11). *Example—II*: Let the rotation angle be radians be represented in 16-bit fixed point format as 0010_1000_0011_0111. The is 13. As which is the elementary angle used will correspond to . Also from Table I, it can be observed that the elementary angle required to realize this rotation angle is 0.125 radians with shift-index . The pseudo-code for generating the micro-rotation sequence is given by algorithm-1 for .With slight modifications a similar algorithm can be designed for .

Algorithm 1 Micro-Rotation Sequence Generation

```
Input: Angle to be rotated $\theta_i$
$MSO_{location}(ML)$ = MSO – LI output when input is $\theta_i$
if $ML = 15$ then
    $\alpha = 0.25$ radians
    Shift $s_i = 2$
    $\theta_{i+1} = \theta_i - \alpha$
else
    Shift $s_i = 16 - ML$
    $\theta_{i+1} = \theta_i$ with $\theta_i[ML]$ = ‘0’
end if
```

convergence (RoC) is an important aspect in the design of the CORDIC algorithm and determines the scope of the algorithm. To determine the RoC we first ensure the number of iterations to limit the latency of implementation of the algorithm, which determines the number of iterations used for realizing CORDIC rotations.

A. Determination of Number of Iterations

The maximum angle that can be handled by the micro-rotation sequence generation block is fixed to . For realizing any angle of rotation lying in the range maximum of three iterations of the basic-shift are required; similarly, for basic-shift maximum of six iterations of basic-shift are required. Therefore, in the design of micro-rotation sequence generator for and for The rest iterations affect the accuracy, hence we simulate the design for various values of . The error is tabulated in Table II. With maximum of iterations, the maximum percentage error in cosh and sinh values is 0.1% and 0.5% respectively for basic-shift 2, and 0.04% and 0.12% respectively for basic-shift 3.

B. Extension of the RoC

The basic RoC for the proposed CORDIC is as the maximum angle handled by the micro-rotation sequence generator is . We can extend the RoC to by storing some known values of exponents and sine and cosine hyperbolic functions in a ROM, at the octant boundaries. The mathematical identities used for extending the RoC are (14); the values

TABLE III
INITIAL VALUES FOR EXPONENT COMPUTATIONS

Rotation Angle	LUT Values	
	X_0	Y_0
$[0, \pi/4]$	1	0
$[\pi/4, \pi/2]$	$e^{\pi/2}$	0
$[\pi/2, 3\pi/4]$	$e^{\pi/2}$	0
$[3\pi/4, \pi]$	e^{π}	0
$[0, -\pi/4]$	1	0
$[-\pi/4, -\pi/2]$	$e^{-\pi/2}$	0
$[-\pi/2, -3\pi/4]$	$e^{-\pi/2}$	0
$[-3\pi/4, -\pi]$	$e^{-\pi}$	0

TABLE IV
INITIAL VALUES FOR HYPERBOLIC COMPUTATIONS

Rotation Angle	LUT Values	
	X_0	Y_0
$[0, \pi/4]$	1	0
$[\pi/4, \pi/2]$	$\cosh \pi/2$	$-\sinh \pi/2$
$[\pi/2, 3\pi/4]$	$\cosh \pi/2$	$\sinh \pi/2$
$[3\pi/4, \pi]$	$\cosh \pi$	$-\sinh \pi$
$[0, -\pi/4]$	1	0
$[-\pi/4, -\pi/2]$	$\cosh \pi/2$	$-\sinh \pi/2$
$[-\pi/2, -3\pi/4]$	$\cosh \pi/2$	$\sinh \pi/2$
$[-3\pi/4, -\pi]$	$\cosh \pi$	$-\sinh \pi$

$$e^{\theta} = \cosh\theta + \sinh\theta$$
$$e^{\theta} = e^{\beta+\gamma} = e^{\beta}\cdot(\cosh\gamma + \sinh\gamma) \qquad (14a)$$
$$\cosh\theta = \cosh(\beta+\gamma) = \cosh\beta\cdot\cosh\gamma + \sinh\beta\cdot\sinh\gamma$$
$$\sinh\theta = \sinh(\beta+\gamma) = \sinh\beta\cdot\cosh\gamma + \cosh\beta\cdot\sinh\gamma \qquad (14b)$$

The ROMrequirement of the proposed CORDIC processor is when implemented for exponent calculations, while it is when implemented for computing the hyperbolic functions. The RoC can be manipulated to either by changing the values of the ROM.

5. ERROR ANALYSIS

The error analysis of any CORDIC algorithm consists of two parts: (i) residue angle error, and (ii) error in the coordinate values. In previous sections, the combined effect of various approximations on coordinate values is discussed. In this section, we present the errors in coordinate due to residue angle error and rotation matrix approximation.

A. Residual Angle Error

In the proposed methodology, desired angle of rotation is expressed as:

$$\theta = \sum^{\text{for } n \text{ iterations}} 2^{-s_i}, \quad s_{basic} \le s_i \le (N-1)$$
$$\text{where } s_{basic} \text{ is basic} - \text{shift}, \ N \text{ is word} - \text{length} \quad (15)$$

We identify the micro-rotations by using the bit-representation of the desired rotation angle. The residue angle error depends on the number of bits set in the radix-2 representation of the rotation angle, and varies for different rotation angles.

Therefore, we derive the worst-case angle error in the range of convergence . The maximum number of iterations are fixed for all rotation angles, i.e., eight(eleven) for basic-shift 2(3). When the input rotation angle has the MSB-nibble value 4'b1011, four(six) iterations are required for the basic-shift ; while, three(five) or less iterations are required in case of other MSB-nibble values. From second MSB-nibble onwards, irrespective of the basic-shift, each bit set to 1'b1 in the radix-2 representation of the rotation angle would require one iteration; maximum four iterations are required if the second MSB nibble value is 4'b1111. For basic-shift , the iteration count is eight, hence the worst-case error is . For basic-shift , iteration count is eleven, this allows one iteration for third MSB-nibble, therefore the worst-case error is . This worst-case residue angle error is specific to the rotation angle of 16'b1011_1111_1111_1111, while for other rotation angles the residue angle error would be less. In the proposed 16-bit fixed point representation scheme, 16'b1011_1111_1111_1111 is 42.97 degrees; the worst-case residue angle error is 0.2229 degrees with basic-shift 2, and 0.1110 degrees with basic-shift 3.

B. Coordinate Error: Effect of Residual Angle

The maximum angle deviation in the proposed scheme is 0.2229 degrees and 0.111 degrees for basic-shift 2 and 3, respectively. In the worst case scenario, the final rotation realized will be radians or . In this case, the coordinates values computed are:

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} \cosh(\theta-\phi) & \sinh(\theta-\phi) \\ \sinh(\theta-\phi) & \cosh(\theta-\phi) \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$\text{where } \phi = \begin{cases} \pi/807 & s_{basic}=2 \\ \pi/1621 & s_{basic}=3 \end{cases} \qquad (16)$$

The error in the coordinate values is:

$$\begin{aligned} x_{error} &= (x_0 \cosh\theta + y_0 \sinh\theta)(\cosh\phi - 1) \\ &\quad - (x_0 \sinh\theta + y_0 \cosh\theta)\sinh\phi \\ y_{error} &= (y_0 \cosh\theta + x_0 \sinh\theta)(\cosh\phi - 1) \\ &\quad - (y_0 \sinh\theta + x_0 \cosh\theta)\sinh\phi \qquad (17) \end{aligned}$$

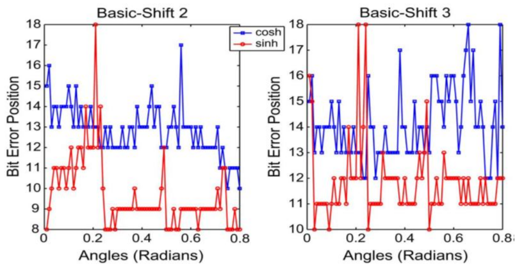


Figure. 4. Bit-error-position plot for cosh and sinh values calculated using proposed algorithm.

C. Coordinate Error: Effect of Rotation Matrix Approximation

In the proposed methodology using the rotation matrix of (9), the error in the coordinate values for one iteration is as:

$$\begin{aligned} x_{error} &= x_0\left(1 + 2^{-(2s_i+1)} - \cosh\theta\right) \\ &\quad + y_0\left(2^{-s_i} + 2^{-(3s_i+2)} - \sinh\theta\right) \\ y_{error} &= y_0\left(1 + 2^{-(2s_i+1)} - \cosh\theta\right) \\ &\quad + x_0\left(2^{-s_i} + 2^{-(3s_i+2)} - \sinh\theta\right) \qquad (18) \end{aligned}$$

6. WAVEFORM GENERATION

In this section, we propose a new technique for generating arbitrary waveforms using the scale-free hyperbolic CORDIC processor. In the proposed design, the phase of hyperbolic DDS function generator is randomly modulated using a linear feedback shift register (LFSR). In the next subsection, we discuss the design of hyperbolic DDS based on the proposed CORDIC processor, following which the architecture of proposed AWG is described. The proposed DDS can be used to generate hyperbolic, exponential and other arbitrary waveforms. For generating sinusoidalwaveforms, the output of circular CORDIC processor replaces the hyperbolic CORDIC processor in the proposed DDS.

A. Proposed Hyperbolic DDS

The block diagram of the proposed hyperbolic DDS is shown in Fig. 5. The proposed DDS consists of a counter, multiplication unit, a LUT and scale-free hyperbolic CORDIC processor.

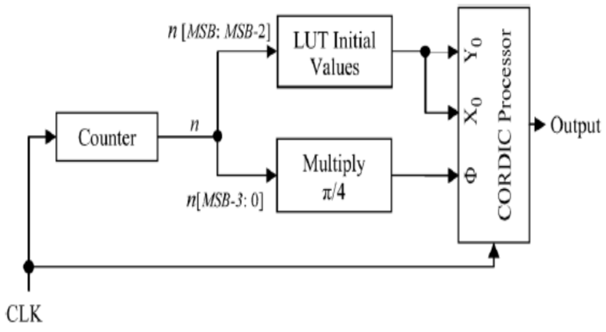


Figure. 5. Proposed hyperbolic DDS architecture.

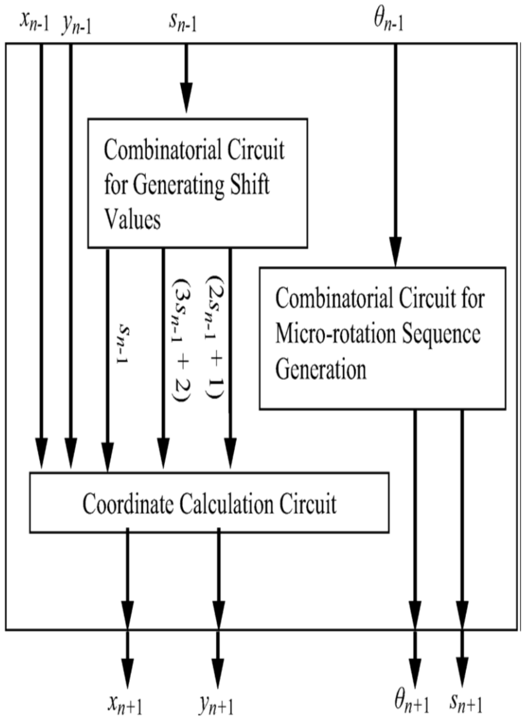

x_{n-1}
y_{n-1}
s_{n-1}
θ_{n-1}
Combinatorial Circuit for Generating Shift Values
s_{n-1}
$(3s_{n-1}+2)$
$(2s_{n-1}+1)$
Combinatorial Circuit for Micro-rotation Sequence Generation
Coordinate Calculation Circuit
x_{n+1}
y_{n+1}
θ_{n+1}
s_{n+1}

Figure. 6. Structure of pipelined stage of proposed scale-free hyperbolic CORDIC processor.

The counter output is used to generate a linear ramp signal which is used subsequently as the phase of the hyperbolic function, such that, the phase of hyperbolic function changes periodically from to With each increment of the counter, the phase is incremented by:

$$\Delta\phi = \frac{2\pi}{2^N} \qquad (19)$$

The counter output ‘ ’, thus corresponds to the phase:

$$\phi = \frac{2\pi \times n}{2^N} - \pi \qquad (20)$$

The hyperbolic CORDIC processor maps the phase to The proposed scale-free hyperbolic CORDIC processor has eight (or eleven) identical pipelined stages for basic-shift 2 (or 3), where each stage performs a CORDIC iteration based on the proposed CORDIC algorithm. The structure of each stage is shown in Fig. 6. It consists of three computing blocks namely, (i) micro-rotation sequence generator, (ii) shift-value estimator, (iii) coordinate calculation unit. The combinatorial circuit for generating the micro-rotation sequence is shown in Fig. 7. The coordinate calculation unit based on rotation matrix (12) is shown in Fig. 8. The shift values required for coordinate calculations are obtained from the circuit shown in Fig. 9. The DDS generates the phase which varies linearly with the counter. The three Most-Significant-Bits (MSB) of the counter divide the entire coordinate space into octants as shown in Fig. 10. These three MSBs are used as the address of the

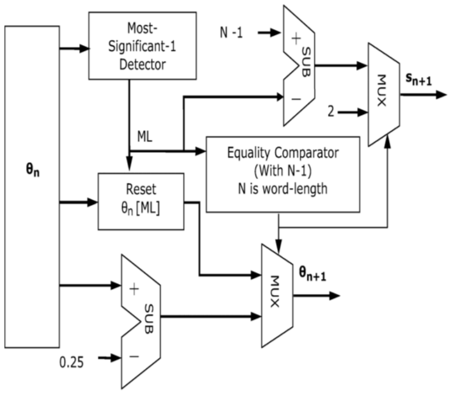

Most-Significant-1 Detector
N -1
SUB
MUX
s_{n+1}
2
ML
Equality Comparator (With N-1) N is word-length
θ_n
Reset θ_n [ML]
θ_{n+1}
0.25

Figure. 7. Micro-rotation sequence generation.

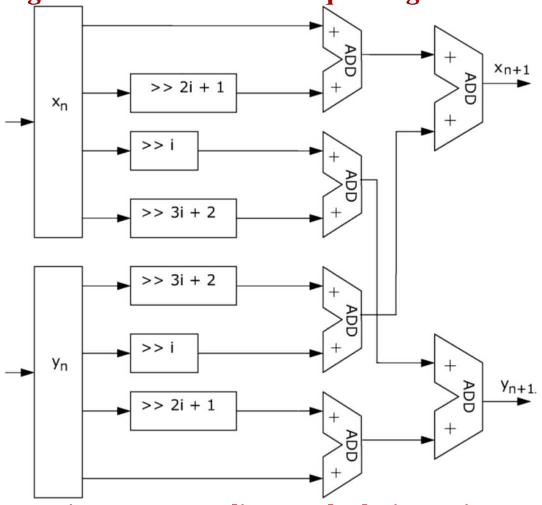


Figure. 8. Coordinate calculation unit.

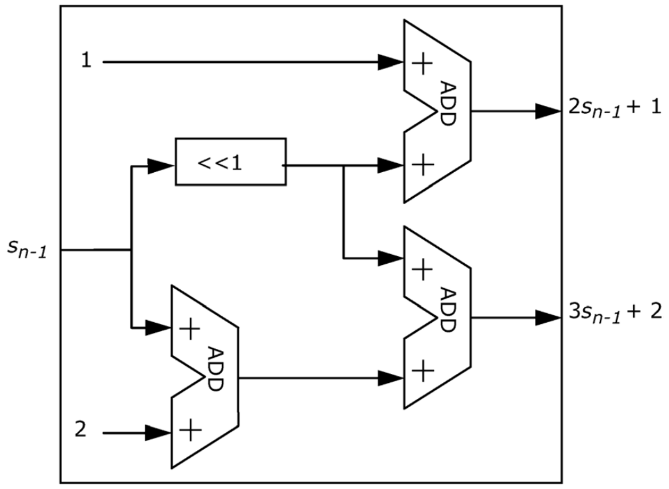


Figure. 9. Shift value estimation.

LUT that stores the initial values for the CORDIC processor, while the rest of the bits are multiplied by to generate the

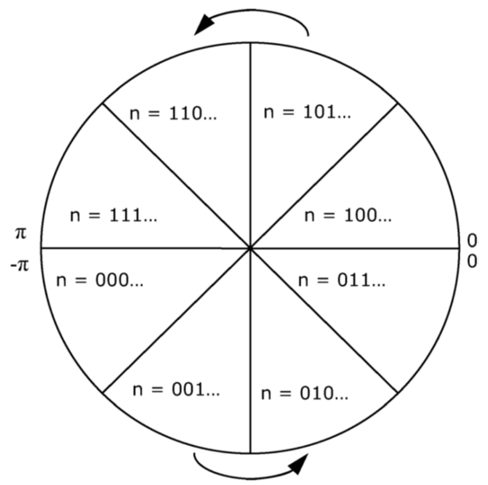


Figure. 10. Distribution of over the coordinate space.

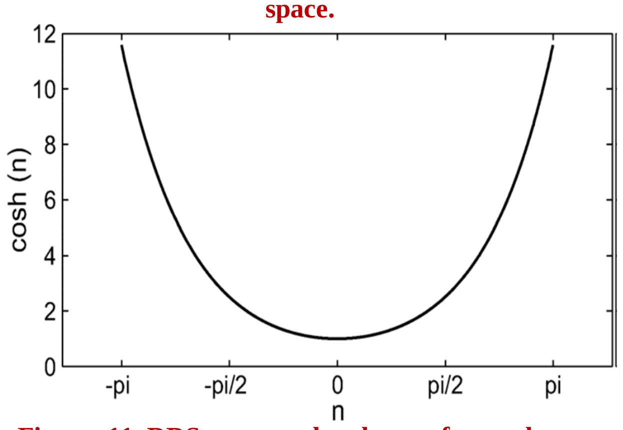


Figure. 11. DDS generated cosh waveform. phase, which always lies between .

The multiplication by is performed by a multiplier consisting of an optimized shift-add tree. The periodic cosh waveform generated by the proposed DDS architecture is shown in Fig. 11. Apart from hyperbolic waveforms, the proposed DDS architecture can be modified to generate exponential waveforms as well. To generate exponential-up/down waveform, the output in Fig. 5 is summation of -coordinate and -coordinate outputs of the CORDIC processor. The exponential-up signal is generated using an up-counter, while for an exponential-down signal a down-counter is used. The periodic exponential-up signal generated using the proposed hyperbolic DDS architecture is shown in Fig. 12.

B. Random Phase Modulated Waveform Generator Using the Proposed Hyperbolic DDS

The proposed DDS architecture can be modified to generate random phase modulated waveforms. To reduce the hardware cost, instead of replicating the DDS multiple times, we randomly modulate the phase of a single DDS for generating random modulated waveforms. The block diagram for the proposed design for random modulated waveform generation is shown in Fig. 13. A random number generated by LFSR is added with the counter of DDS to modulate its phase. The counter increments linearly from 0 to rollover count . Once the counter value exceeds , it is reset

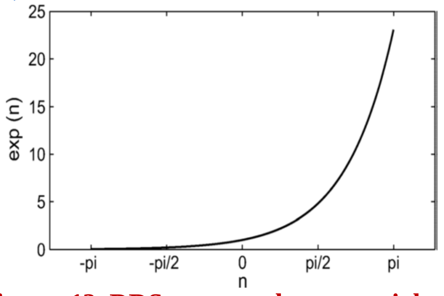


Figure. 12. DDS generated exponential-up waveform.

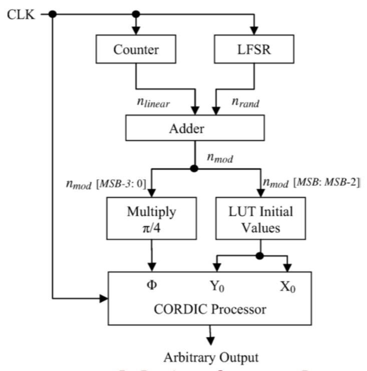


Figure. 13. Proposed design for random modulated waveform generation.

to zero. The counter output after ' ' clock cycles is given by (21), where is the linear increments of the counter, such that:

$$n_{linear} = n \cdot c_{step} \quad (21)$$

The modulated phase is given by (22), where the adder ignores the carry generated by the addition of to restrict the modulated phase to .

$$n_m = (n \cdot c_{step} + n_{rand}) \bmod 2^N \quad (22a)$$

$$\phi_m = \frac{2\pi \times n_m}{2^N} - \pi \quad (22b)$$

Equation (22b) shows a linear relation between and . The threeMSBs of partition the whole coordinate space into octants, and follow the same pattern as counter inDDS, shown in Fig. 10. Therefore, the octants are identified by threeMSBs of . The rest of the bits of are multiplied by to generate the phase which always lies in the range . The initial values required to be stored in an LUT are given in Table V. The modulated phase lies in the range , so the output amplitude varies arbitrarily in the range [1, 11.59]. It can be

Table V Initial values for realizing modulated waveforms

$n_m[MSB : MSB-2]$	LUT Values	
	X_0	Y_0
000 111	$\cosh 3\pi/4$	$\sinh 3\pi/4$
001 110	$\cosh \pi/2$	$\sinh \pi/2$
010 101	$\cosh \pi/4$	$\sinh \pi/4$
011 100	1	0

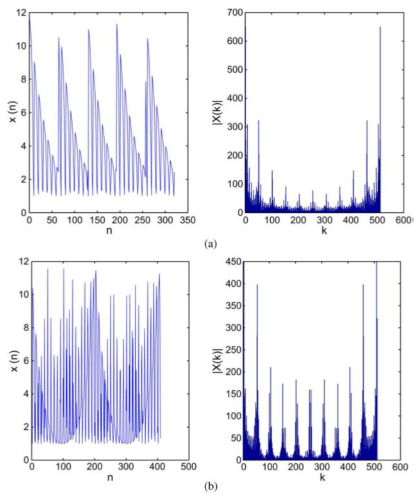


Figure 14. Random modulated waveforms generated using proposed waveform generator.

extended to [1, 267.74] by changing the range of the modulated phase to . By selecting the initial seed value of LFSR, rollover count and counter step, the proposed waveform generator can generate various kinds of modulated/arbitrary waveforms. Two examples of modulated waveforms are shown in Fig. 14.

C. Fourier Analysis of Proposed Waveform Generator

The proposed AWG generates samples of random modulated waveforms whose phase is defined by (22b). The signal generated by the waveform generator can now be defined as:

$$x(n) = \cosh(l \cdot n_m - \pi) \quad \text{where } l = \frac{2\pi}{2^N} \quad (23a)$$

$$x(n) = C_1 \cdot e^{l \cdot n_m} + C_2 \cdot e^{-l \cdot n_m},$$

$$\text{where } C_1 = \frac{e^{-\pi}}{2} \ \& \ C_2 = \frac{e^{\pi}}{2} \quad (23b)$$

The waveforms generated by the proposed AWG are periodic in nature. The period of the arbitrary waveform is a function of counter period and LFSR period (24). The depends on the polynomial used to implement the LFSR. The coefficients of the polynomial of the LFSR belong to binary Galois field GF(2). The polynomial which we have used to implement LFSR is . According to the Appendix C, is given by:

$$T_{LFSR} = 10 \quad \text{and} \quad T_{cnt} = \left\lceil \frac{RO_{cnt}}{c_{step}} \right\rceil$$

$$T_{AWG} = LCM(T_{cnt}, T_{LFSR}) \quad (24)$$

The Discrete Fourier transform (DFT) of the proposed waveform generator signal (23b) is given by:

$$X(k) = C_1 \sum_{n=0}^{M-1} \left(e^{l \cdot n_m} \cdot e^{-j2\pi nk/M}\right) + C_2 \sum_{n=0}^{M-1} \left(e^{-l \cdot n_m} \cdot e^{-j2\pi nk/M}\right)$$

$$\text{where } k = 0, 1, 2, \ldots, M-1 \text{ and } M \text{ is period of waveform} \quad (25a)$$

From (25a),we observe the frequency components of the generated waveforms depend on . The modulated depends on counter step and the LFSR value.

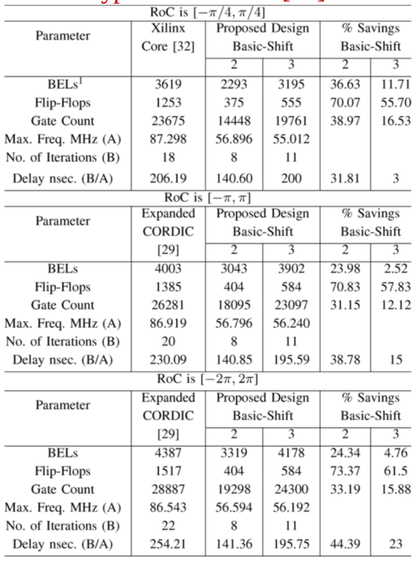
7. FPGA IMPLEMENTATION RESULTS AND COMPLEXITY ISSUES

The proposed AWGis coded in Verilog and synthesized using Xilinx ISE 9.2i to bemapped in to Xilinx Spartan 2E (xc2s200e- 6pq208) device. For a 16-bit output width, the proposed AWG implemented using basic-shift 2, in the proposed hyperbolic CORDIC processor, consumes 1076 slices and 2016 4-input LUTs. It achieves a maximum frequency of 56.383 MHz, with a gate count of 17580. The proposed AWG designed with basicshift 3 on the same device provides maximum operating frequency of 55.661 MHz and consumes 1340 slices and 2499. 4-input LUTs with 22294 gate count.

A. Complexity Consideration: Proposed Hyperbolic CORDIC Processor

We compare the proposed hyperbolic CORDIC processor with the Xilinx CORDIC Core v 3.0 [32] and expanded CORDIC [29] in terms of area and time complexities in Table VI. The implementation of Xilinx Core [32] is based on conventional CORDIC algorithm, designed for the same bit-accuracy as that obtained by the proposed CORDIC algorithm. The Xilinx Core is optimized for implementing hyperbolic functions only. The elementary rotation in hyperbolic coordinates do not converge in conventional CORDIC, and convergence is achieved only by repeating certain micro-rotations [33]; for 16-bit implementation, iterations and 13 need repetition. Hence, the total iteration count is 18 with the bit-error-position of minimum 8-10-bits [34]. In [29], the RoC of conventional hyperbolic CORDIC processor is expanded

Table VI complexity comparison: proposed hyperbolic cordic processor With xilinx cordic core and expanded hyperbolic cordic [29]



RoC is $[-\pi/4, \pi/4]$

Parameter	Xilinx Core [32]	Proposed Design Basic-Shift		% Savings Basic-Shift	
		2	3	2	3
BELs[1]	3619	2293	3195	36.63	11.71
Flip-Flops	1253	375	555	70.07	55.70
Gate Count	23675	14448	19761	38.97	16.53
Max. Freq. MHz (A)	87.298	56.896	55.012		
No. of Iterations (B)	18	8	11		
Delay nsec. (B/A)	206.19	140.60	200	31.81	3

RoC is $[-\pi, \pi]$

Parameter	Expanded CORDIC [29]	Proposed Design Basic-Shift		% Savings Basic-Shift	
		2	3	2	3
BELs	4003	3043	3902	23.98	2.52
Flip-Flops	1385	404	584	70.83	57.83
Gate Count	26281	18095	23097	31.15	12.12
Max. Freq. MHz (A)	86.919	56.796	56.240		
No. of Iterations (B)	20	8	11		
Delay nsec. (B/A)	230.09	140.85	195.59	38.78	15

RoC is $[-2\pi, 2\pi]$

Parameter	Expanded CORDIC [29]	Proposed Design Basic-Shift		% Savings Basic-Shift	
		2	3	2	3
BELs	4387	3319	4178	24.34	4.76
Flip-Flops	1517	404	584	73.37	61.5
Gate Count	28887	19298	24300	33.19	15.88
Max. Freq. MHz (A)	86.543	56.594	56.192		
No. of Iterations (B)	22	8	11		
Delay nsec. (B/A)	254.21	141.36	195.75	44.39	23

BEL is Basic Element of Logic, includes MUXes and LUTs required by the design. Typically BEL corresponds 1:1 with an instance in the logical/structural view of the design. The BEL consumption of the circuit remains unchanged with the change in FPGA device. Therefore, it is better measure to compare area than slices, as number of slices change with change in FPGA device. using additional iterations. For RoC of , it requires two additional iterations apart from usual conventional CORDIC iterations, which increases the total iteration count to 20. Hence, the latency of the CORDIC algorithm of [29] is significantly higher than the proposed CORDIC algorithm which involves only 8 iterations. The proposed CORDIC can also work with the latency of 8 iterations for the RoC of . This RoC can be realized by using a small circuit comprising of a complementor and a 2 1 multiplexer with the pre-processing unit (shown in Fig. 15). To achieve this RoC, the algorithm of [29] would require 22 iterations.

B. Complexity Considerations of Waveform Generator

We compare the area and time complexities of the proposed AWG with existing CORDIC-based AWGs in Table VII and Table VIII, respectively. The proposed AWG on an average requires 36.5% less area as compared to existing CORDIC-based AWGs. While it has approximately 5.7% less throughput than Xilinx-CORDIC Core based AWG, it has 7.3% higher throughput than scaling-free CORDIC [22]-based AWG. The

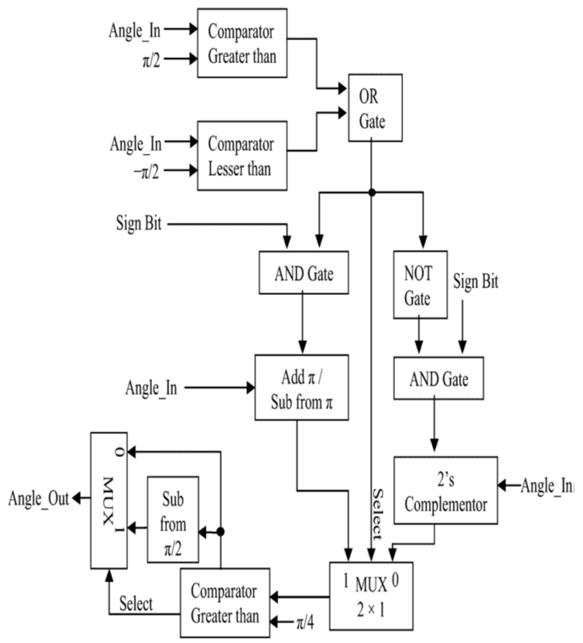


Figure. 15. Pre-processor unit to extend the RoC of the proposed scale-free hyperbolic CORDIC.

Table VII Area Comparison: Proposed Waveform Generator With Other CORDIC Based AWGS

Conventional AWG[1]			Proposed AWG			% Savings	
No. DDS	CORDIC Algo	Gate Count	No. DDS	Gate Count			
				2	3	2	3
2	Xilinx [32]	40244	1	17580	22294	56.31	44.6
	Scale-Free [22]	25353				30.65	12.06
	E.S.F.[23]	16226					
3	Xilinx [32]	60366	1	17580	22294	70.87	63.06
	Scale-Free [22]	38030				53.77	41.37
	E.S.F.[23]	24340				27.77	08.40
4	Xilinx [32]	80488	1	17580	22294	78.15	72.30
	Scale-Free [22]	50707				65.33	56.03
	E.S.F.[23]	32452				45.82	31.30

The conventional CORDIC gate-counts are calculated using Xilinx CORDIC-IP. The scale-free and E.S.F. CORDIC architectures consume 36% and 37% respectviely less area than conventional CORDIC. Therefore, their gate-counts are 36% and 37% less than the conventional CORDIC gate-count. The total gate-count of the CORDIC-based AWGs is higher as it includes DDS implementation circuitry like counter, phase angle generator.

Table VIII Time Comparison: Proposed Waveform Generator With Other Cordic Based Awgs

CORDIC-based AWG		Latency	Max. Clock Period (MHz)[1]
Conventional	Xilinx [32]	16	59.791
	Scale-Free[22]	12	52.54
	E.S.F.[23]	9	56.16
Proposed	$s_{basic} = 2$	8	56.383
	$s_{basic} = 3$	11	55.661

Maximum operating frequency as obtained on Xilinx Spartan 2E (xc2s200e-6pq208) device. latency of the proposed AWG using is lowest as it requires only 8 pipeline stages.

8. CONCLUSION

A scale-free hyperbolic CORDIC algorithm is proposed and used for arbitrary waveform generation. The key features of the proposed algorithm are that it is completely scaling-free, provides greater RoC, and reduces the number of iterations. A generalized micro-rotation scheme based on most-significant-1 detector with single direction micro-rotations is used to eliminate the redundant CORDIC iterations.Using the proposed CORDIC algorithm, we have suggested a low complexity waveform generator using a single DDS by random phase modulation. The proposed AWG requires on an average 36% less area, and less latency compared with existing CORDIC-based designs with nearly the same throughput rate.

APPENDIX A

Lemma: The order of approximation of Taylor series expansion of hyperbolic sine and cosine functions determines the highest elementary angle to be used for CORDIC iterations. *Proof:* For and approximation order the highest term neglected in sinh series and cosh series are defined as:

$$T_{Sn} = \frac{2^{-(n+c_1)i}}{(n+c_1)!}, \quad c_1 = \begin{cases} 1 & \text{for even } n \\ 2 & \text{for odd } n \end{cases}$$
$$T_{Cn} = \frac{2^{-(n+c_2)i}}{(n+c_2)!}, \quad c_2 = \begin{cases} 2 & \text{for even } n \\ 1 & \text{for odd } n \end{cases} \qquad (26)$$

The values of iteration indices used in CORDIC operations are chosen such that, the terms neglected in the Taylor series get a right shift greater than the word-length (-bits). This causes them to be zero and their role in calculating values is obviated. The lowest degree term neglected in the series decides the basic-shift value imposing a restriction on the smallest elementary angle used. Hence, decides the basic-shift and lays the foundation for the first CORDIC iteration. The lowest degree term neglected for approximation order is given by .

$$T_n = 2^{-[(n+1)i+\log_2(n+1)!]} \qquad (27)$$

Consequently, for to be zero for a word-length of -bits, the
condition is:

$$[(n+1)i + \log_2(n+1)!] \geq b$$
$$\Rightarrow i \geq \left\lceil \frac{b - \log_2(n+1)!}{(n+1)} \right\rceil \qquad (28a)$$
$$s_{basic} = \left\lceil \frac{b - \log_2(n+1)!}{(n+1)} \right\rceil \qquad (28b)$$

APPENDIX B

Lemma: Reiteration of elementary angle is applicable only for the basic-shift; all other values of shifts are non-repetitive. *Proof:* Let the start shift be represented as ‘ ’ and elementary angles as , ‘ ’ micro-rotations of elementary angle , are represented as:

$$N \cdot \alpha_p = (N \bmod 2) \cdot \alpha_p + \left\lfloor \frac{N}{2} \right\rfloor \cdot \alpha_{p-1} \qquad (29)$$

The minimum number of iterations required for realizing, micro-rotations of elementary angle are:

$$N \cdot \alpha_p = \sum_{k=0}^{p-(start)-1} \left(\left\lfloor \frac{N}{2^k} \right\rfloor \bmod 2 \right) \cdot \alpha_{p-k} + \left(\left\lfloor \frac{N}{2^{p-(start)}} \right\rfloor \bmod 2 \right) \cdot \alpha_{max} \qquad (30)$$

APPENDIX C

Lemma: Every polynomial with coefficients in GF(2) having divides for some . The smallest for which this is true is called the period of *Proof:* The proof is out of scope of the paper.

REFERENCES

[1] D. Peterson and B. Precision, *Function Generator and ArbitraryWaveform Generator Guidebook*. Yorba Linda, CA: B&K Precision Corporation, 2010.

[2] TTI, Technical Specifications DDS Function/Arbitrary Generator.

[3] W. Hu, C. L. Lee, and X.Wang, “Arbitrary waveform generator based on direct digital frequency synthesizer,” in *Proc. 4th IEEE Symp. Electronic Design, Test and Applications*, Jan. 2008, pp. 567–570.

[4] A. Ashrafi, R. Adhami, L. Joiner, and P. Kaveh, “Arbitrary waveform DDFS utilizing Chebyshev polynomials interpolation,” *IEEE Trans. Circuits Syst. I*, vol. 51, no. 8, pp. 1468–1475, Aug. 2004.

[5] J. Yang and T. Sarkar, “A new doppler-tolerant polyphase pulse compression codes based on hyperbolic frequency modulation,” in *Proc. IEEE Conf. Radar*, Jun. 2007, pp. 265–270.

[6] J. Yang and T. Sarkar, “Doppler-invariant property of hyperbolic frequency modulated waveforms,” *Microw. Opt. Technol. Lett.*, vol. 48, p. 11741179, 2006.

[7] D. D. Caro, N. Petra, and A. G. M. Strollo, “A 380 MHz direct digital synthesizer/mixer with hybrid CORDIC architecture in 0.25 um CMOS,” *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 151–160, Jan. 2007.

[8] S. F. Hsiao, Y. H. Hu, and T. B. Juang, “A memory efficient and high speed sine/cosine generator based on parallel CORDIC rotations,” *IEEE Signal Process. Lett.*, vol. 11, no. 2, pp. 152–155, Feb. 2004.

[9] P. P. Vaidyanathan, “A unified approach to orthogonal digital filters and wave digital filters based on the LBR two-pair extraction,” *IEEE Trans. Circuits Syst. I*, vol. CAS-32, pp. 673–686, Jul. 1985.

[10] M. Despain, “Fourier transform computers using CORDIC iterations,” *IEEE Trans. Comput.*, vol. 23, pp. 993–1001, Oct. 1974.

[11] K. Maharatna and S. Banerjee, “A VLSI array architecture for Hough transform,” *Pattern Recognit.*, vol. 34, pp. 1503–1512, 2001.

[12] P. K. Meher *et al.*, "50 years of CORDIC: Algorithms, architectures and applications," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.

[13] B. Gisuthan and T. Srikanthan, "FLAT CORDIC: A unified architecture for high speed generation of trigonometric and hyperbolic functions," in *Proc. 43rd IEEE Midwest Symp. Circuits and Systems*, Lansing, MI, Aug. 2000, pp. 1414–1417.

[14] T. B. Juang, S. F. Hsiao, and M. Y. Tsai, "Para-CORDIC: Parallel CORDIC rotation algorithm," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 8, pp. 1515–1524, Aug. 2004.

[15] C. S. Wu and A. Y. Wu, "Modified vector rotational CORDIC (MVRCORDIC) algorithm and architecture," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 548–561, Jun. 2001.

[16] C. S. Wu, A. Y. Wu, and C. H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," *IEEE Trans. Circuits Syst. II*, vol. 50, no. 9, pp. 589–601, Sep. 2003.

[17] Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithmimplementation," *IEEE Trans. Comput.*, vol. 42, pp. 99–102, Jan. 1993.

[18] C. H. Lin and A. Y. Wu, "Mixed-Scaling-Rotation-CORDIC (MSRCORDIC) algorithm and architecture for high-performance vector rotational DSP applications," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 11, pp. 2385–2396, Nov. 2005.

[19] M. G. B. Sumanasena, "A scale factor correction scheme for the CORDIC algorithm," *IEEE Trans. Comput.*, vol. 57, no. 8, pp. 1148–1152, Aug. 2008.

[20] J. Villalba *et al.*, "CORDIC architecture with parallel compensation of the scale factor," in *Proc. Int. Conf. Application Specific Array Processors*, Strasbourg, France, Jul. 1995, pp. 258–269.

[21] K. Maharatna, A. Troya, S. Banerjee, and E. Grass, "Virtually scaling free adaptive CORDIC rotator," *IEEE Proc. Comp. Dig. Tech.*, vol. 151, no. 6, pp. 448–456, Nov. 2004.

[22] K. Maharatna *et al.*, "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 11, pp. 1463–1474, Nov. 2005.

[23] F. Jaime *et al.*, "Enhanced scaling-free CORDIC," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 7, pp. 1654–1662, Jul. 2010.

[24] B. Gisuthan, T. Srikanthan, and K. V. Ansari, "A high speed flat CORDIC based neuron with multi-level activation function for robust pattern generation," in *Proc. 5th IEEE Workshop on Computer Architectures for Machine Perception*, Padova, Italy, Sep. 2000, pp. 87–94.

[25] M. Qian, "Application of CORDIC algorithm to neural networks VLSI design," in *Multiconf. Computational Engineering in Systems Applications IMACS*, Oct. 2006, pp. 504–508.

[26] A. M. Base, R. Watzel, U. M. Base, and S. Foo, "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, Dec. 2003.

[27] M. Chakraborty, S. Pervin, and T. N. Lamba, "A hyperbolic LMS algorithm for CORDIC based realization," in *Proc. 11th IEEE Signal Processing Workshop on Statistical Signal Processing*, Aug. 2001, pp. 373–376.

[28] A. Boudabous *et al.*, "Implementation of hyperbolic functions using CORDIC algorithm," in *Proc. Int. Conf. Microelectronics*, 2004, pp. 738–741.

[29] D. R. L. Obregon and C. P. A. Rios, "A fixed-point implementation of the expanded hyperbolic CORDIC algorithm," *Latin American Applied Research (online)*, vol. 37, no. 1, pp. 83–91, 2007.

[30] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. 38th Spring Joint Computer Conf.*, Atlantic City, NJ, 1971, pp. 379–385.

[31] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Comput.*, vol. EC-8, pp. 330–334, Sep. 1959.

[32] *"Xilinx Logic Core Product Specifications CORDICv3.0 DS249,"* Xilinx, 2005.

[33] H. M. Ahmed, J. M. Delosme, and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing," *Computer*, vol. 15, no. 1, pp. 65–82, Jan. 1982.