Volume 2 No.6, June 2012 ISSN 2224-3577
International Journal of Science and Technology

http://www.ejournalofsciences.org

Direct Digital Synthesizer using Pipelined CORDIC Algorithm for Software Defined Radio

T.Menakadevi[1], M.Madheswaran[2]
[1]Department of ECE, Adhiyamaan College of Engineering, Hosur-635109, Tamil Nadu, India
[2]Centre for Research and Development, Muthayammal Engineering College, Rasipuram , Namakkal, Tamil Nadu, India

ABSTRACT

This paper proposes Design and Implementation of CORDIC algorithm for Direct Digital Synthesizer. COordinate Rotation DIgital Computer (CORDIC) algorithm is an interesting technique for phase to sine amplitude conversion. The algorithm proposed in this design is to utilize dynamic transformation rather than ROM static addressing. The proposed CORDIC design is based on Pipeline data path Architecture. By using pipeline architecture, the design is able to calculate continuous input, has high throughput, and doesn't need ROM or registers to save constant angle iteration of CORDIC. CORDIC algorithm provides fast and area efficient computations of sine and cosine functions without using ROM LUTs. This paper is focused on the Direct Digital Synthesizer using CORDIC approach, to increase the speed with minimum area requirement in FPGA. To prove the better performance of proposed DDS architecture it is compared favorably with several existing DDS architectures.

KEYWORDS: *FPGA, Direct Digital synthesis, ROMLUT, CORDIC*

1. INTRODUCTION

Direct Digital Synthesis (DDS) is an electronic method for digitally creating arbitrary waveforms and frequencies from a single, fixed source frequency. With the development of VLSI technology and the requirement of modern communication systems, direct digital synthesizers have been widely used in Software Defined Radios and wireless transceivers. A DDS can achieve fast frequency switching in small frequency steps, over a wide band.

In addition, it provides linear phase and frequency shifting with good spectral purity [1].

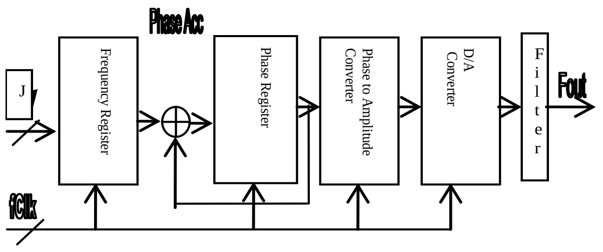


Figure.1: DDS Function Blocks and Signal Flow Diagrams

A DDS is used especially for a precise, high frequency and a phase tunable output.

The basic block diagram of a direct digital frequency synthesizer is shown in Figure.1 [2]. In Figure.1 the main components of a DDS are a phase accumulator, phase-to-amplitude converter (a sine look-up table), a Digital-to-Analog Converter and filter. A DDS produces a sine wave at a given frequency. The frequency depends on three variables; the reference-clock frequency f_{clk} and the binary number programmed into the phase register (frequency control word, M), length of n-bit accumulator. The binary number in the phase register provides the main input to the phase accumulator.

To generate a fixed-frequency sine wave, a constant value the phase increment that is determined by the binary number [M] is added to the phase accumulator with each clock cycle. If the phase increment is large, the phase accumulator will step quickly and thus generate a high frequency sine wave. If the phase increment is small, the phase accumulator will take many more steps, accordingly generating a slower waveform [2, 3]. The digital word in the phase register, M represents the amount; the phase accumulator is incremented each clock cycle. If f_{clk} is the clock frequency, then the frequency of the output sine wave is equal to:

$$f_{out} = \frac{M * f_{clk}}{2^n} \qquad (1)$$

Above equation is known as the DDS "tuning equation." The frequency resolution of the system equals $f_{clk}/2^n$. In a practical DDS system, all the bits out of the phase accumulator are not passed on to the LUT, but are truncated, leaving only the first 13 to 15 MSBs. This reduces the size of the LUT and does not affect the frequency resolution. [4-6].

2. RELATED WORK

Several algorithms are proposed for calculation of sine and cosine function. The digit-by-digit methods for the computation of the elementary functions such as trigonometric, inverse trigonometric, logarithm, exponential, multiplication, and division functions were described by Henry Briggs in 1624 in "Arithmetica Logarithmica" [7]. The programmable CORDIC chip for DSP applications were proposed by D. Timmermann et. in [8]. These are iterative pseudo division and pseudo multiplication processes, which resemble repeated-addition multiplication and repeated-subtraction division. In 1959, Volder has proposed a special purpose digital computing unit known as COordinate Rotation DIgital Computer (CORDIC), while building a real time navigational computer for use in an aircraft [9],[10]. This algorithm was initially developed for trigonometric functions which were expressed in terms of basic plane rotations.

The CORDIC algorithm computes 2D rotation using iterative equations employing shift and add operations. Walther in 1971 [11] has proposed a unified algorithm to compute rotation in circular, linear, and hyperbolic coordinate systems using the same CORDIC algorithm, embedding coordinate systems as a parameter. During the last 50 years of the CORDIC algorithm a wide variety of applications have emerged. The CORDIC algorithm has received increased attention after an unified approach is proposed for its implementation [12].Direct digital frequency synthesis (DDFS) architecture based on the differential CORDIC (DCORDIC) algorithm was presented by Chang Yong Kang et al [13].

The digit-level pipelining in the CORDIC angle path was implemented a two-dimensional systolic array. More recently, the advances in the VLSI technology and the advent of EDA tools have extended the application of CORDIC algorithm to the field of software defined radio, MIMO systems [14] and neural networks [15] etc. Optimized Direct Digital Frequency Synthesizer (DDFS) for complex demodulation using CORDIC was discussed in [16]. DDFS with 8 Hz tuning frequency resolution and 20 bits output data (for sine and cosine waves) was implemented in Xilinx FPGA device giving a maximum operating frequency of more than 306 MHz and a Spurious Free Dynamic Range (SFDR) of 112dBc.

Hybrid COordinate Rotation DIgital Computer (CORDIC) algorithm for designs and implementations of the direct digital frequency synthesizer (DDFS) was proposed [17].Direct digital frequency synthesis (DDFS) based on the Modified Coordinate Rotation (CORDIC) algorithm was proposed in [18]. The technique described in [19] uses Hybrid CORDIC digital synthesizer/mixer corresponds to a rotation of the input vector in the complex plane. This architecture divides the rotation into three sub rotations. The first one uses a few CORDIC stages, in which the rotation directions in parallel computed with the help of a small lookup table. The CORDIC algorithm is employed also in the second sub rotation, where the rotation directions are readily available after a simple recoding of the bits of the residual angle. The final rotation is multiplier based to reduce circuit latency and increase performances. This architecture is implemented in 0.25μm CMOS technology.

Pipelined and parallel implementations of CORDICs for Reconfigurable computing was proposed to achieve very high throughput for rotation, and various other functions such as multiplication, division, as well as hyperbolic and other higher order functions [20]. Sinusoidal signal generation, which is based on the combination of an IIR digital filter and Coordinate Rotations Digital Computer (CORDIC) Arithmetic, was discussed by Chen shijie et al. [21].A CORDIC algorithm for FPGA based computers was extensively investigated by R.Andhraka in [22]. A pre computation based rotation CORDIC algorithm [23] and High speed CORDIC algorithm and architecture for DSP applications [24] were proposed. Redundant CORDIC methods with a constant scale factor for sine and cosine computation is proposed by N. Takagi in [25]. Low Latency Redundant CORDIC is proposed to reduce the latency of redundant CORDIC [26]. Pipelining flat CORDIC based trigonometric function generators were presented by B. Gisuthan et al [27]. In [28], Para-CORDIC algorithm is proposed to precompute the direction of rotations without using ROM. Semi-flat architecture for high speed and reduced area CORDIC chip was investigated in [29]. Iterative based computation eliminates the polarity of micro rotations in CORDIC based sine-cosine generators is proposed in [30]. Fast VLSI implementation of CORDIC algorithm[31] and Double step branching CORDIC algorithm for fast sine and cosine generation was presented in [32]. The wave pipelining technique for CMOS VLSI circuits and FPGA implementation of CORDIC algorithm are deeply discussed by K.K.Parhi in [33]. Fast binary sine/cosine generator [34] and Pipelined CORDIC architecture for the implementation of rotational based algorithm [35] are investigated by P. W. Baker and Y.H.Hu.

This paper is organized as follows. Section.3 presents CORDIC overview. In Section.4 describes the implementation of pipelined CORDIC architecture suitable for reconfigurable hardware. Section.5 deals with simulation results and discussions of proposed CORDIC. Finally the conclusion is summarized in section.6.

3. CORDIC AN OVERVIEW

COordinate **R**otation **DI**gital **C**omputer) is a simple and efficient algorithm to calculate hyperbolic and trigonometric functions. 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, binary shift and table lookup. Volder's algorithm [8] is derived from the general equations for a vector rotation.

$$x' = x\cos(\Phi) - y\sin(\Phi) \quad (2)$$
$$y' = y\cos(\Phi) + x\sin(\Phi) \quad (3)$$

Volder observed that by factoring out a *cos Φ* from both the sides, resulting equation be in terms of the tangent of the angle Φ, the angle of which to find *sin* and *cos*. Next if it is assumed that the angle Φ' is being an aggregate of small angles, and composite angles is chosen such that their tangents are all inverse powers of two, then this equation can be rewritten as an iterative formula which rotates a vector in a Cartesian plane by the angle Φ. These can be rearranged so that,

$x' = \cos(\Phi)[x - y\tan(\Phi)]$ (4)
$y' = \cos(\Phi)[y + x\tan(\Phi)]$ (5)

$z' = z \pm \Phi$ here Φ is the angle of rotation (± sign is showing the direction of rotation) and z is the argument. For the ease of calculation here only rotation anticlockwise direction is observed first. Rearranging the equation (4) and (5)

$x_{i+1} = \cos(\Phi)\,[(x_i - y\tan(\Phi)]$ (6)

$y_{i+1} = \cos(\Phi)[y_i + x\tan(\Phi)]$ (7)

The multiplication by the tangent can be avoided if the rotation angles and therefore *tan(Φ)* are restricted so that $\tan(\Phi) = 2^{-i}$. In the digital hardware this denotes a simple shift operation. Furthermore, if those rotations are performed iteratively and in both directions every value of *tan(Φ)* is represent able. With $\Phi = \arctan(2^{-i})$ the cosine term could also be simplified and since $\cos(\Phi) = \cos(-\Phi)$ it is a constant for a fixed number of iterations. This iterative rotation can be expressed as:

$X_{i+1} = k_i[x_i - yi.di.2^{-i}]$ (8)

$Y_{i+1} = k_i[y_i - xi.di.2^{-i}]$ (9)

$$\begin{aligned} Zi+1 &= Zi - di\Phi \\ &= Z_i - d_i \arctan(2^{-i}) \end{aligned}$$

Where 'i' denotes the number of rotation required to reach the required angle of the required vector $k_i = \cos(\tan^{-1}(2^{-i})) = 1/\sqrt{1+2^{-2i}}$ (10)

and

$d_i = \pm 1.$

Removing the scale constant from the iterative equation yields a shift and add algorithm for vector rotation. The product of the k_i's can be applied elsewhere in the system or treated as part of a system processing gain.[22] k_i is the gain and its value changes as the number of iteration increases. For 8-bit hardware CORDIC approximation method the value of 'k_i' as

$$k_i = \prod_{i=0}^{7} \cos\emptyset i = \cos\emptyset 0.\cos\emptyset 1 \ldots\ldots \cos\emptyset 7$$

$$= \cos 45°\cos 26.565° \ldots\ldots\ldots\ldots\ldots \cos 0.4469°.$$

$$= 0.6703$$

The exact gain depends on the number of iterations and obeys the following relation:

$A_n = \prod_n \sqrt{1+2^{-2i}}$ (11)

Φ is the angle of rotation here for n times rotation. In rotation mode CORDIC equations are,

$X_{i+1} = x_i - yi.di.2^{-i}$ (12)

$Y_{i+1} = y_i - xi.di.2^{-I}$ (13)

$Zi+1 = Z_i - d_i \arctan(2^{-i})$ (14)

Where $d_i = -1$ if $Z_i < 0$, +1 otherwise

This provides the following results,

$X_n = A_n[x_i\cos Z_i - y_i\sin Z_i]$ (15)

$Y_n = A_n[y_i\cos Z_i + x_i \sin Z_i]$ (16)

$Z_i = 0$

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

The series of iterations are required to perform to get an expected angle and in this design the number of iterations are i = 8 and in every iteration the new values of x, y and z depend upon the previous values of the same. This Φ_i is stored in the ROM hardware of the CORIC hardware as a look up table.

4. IMPLEMENTATION OF PIPELINE CORDIC ARCHITECTURE

The CORDIC method can be employed in two different modes: the "rotation" mode and the vectoring" mode. In the rotation mode, the algorithm basic idea consists in decomposing rotation operation into successive basic rotations. Each basic rotation can be realized by shifting and adding shift and add arithmetic operations. The rotation mode of the CORDIC algorithm could be used to compute sine and cosine of an angle θ. Outputs after "n" iterations are computed according to the CORDIC algorithm. The computation of sin (Φ) and cos (Φ) is based on the rotation of an initial vector of unit length, that is aligned with the abscissa (x0=1, y0=1). Moreover, the accumulated angle is initialized with the desired rotation angle. For each iteration, comparison is done between the initial angle and the resulting angle. Then, the comparison sign is used to determine the sign of the next rotation [25].

The pipelined CORDIC architecture is taken to implement this design. VHDL code has been developed for pipelined CORDIC and it is simulated by using Xilinx ISE9.2i Tools. The main advantage of the pipelined architecture compared to non pipelined architecture is high throughput due to the hardwired shifts rather than time and area consuming barrel shifters and elimination of ROM. It may be noted that the pipelined architecture offers throughput improvement by a factor of 'n' for n-bit precision at the expense of increasing the hardware by a factor less than 'n'.[26]. First, the shift operations for each step can be performed by wiring the connections between stages appropriately. There is no need for changing constant values and those can therefore be hardwired as well. The purely unrolled design only consists of

Volume 2 No.6, June 2012 ISSN 2224-3577
International Journal of Science and Technology
©2012 IJST. All rights reserved
http://www.ejournalofsciences.org

combinatorial components and computes one sine value per clock cycle.

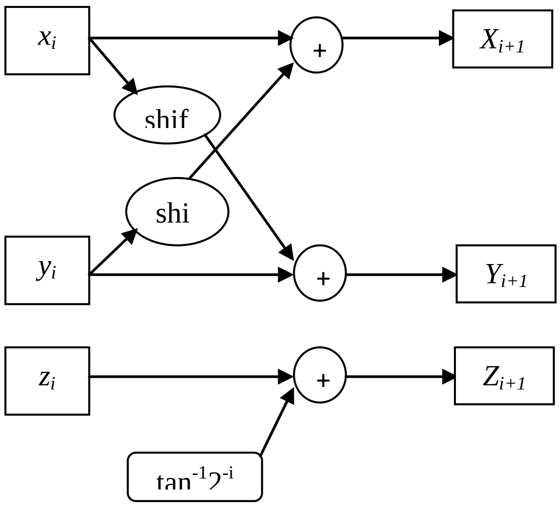


Figure.2: Block diagram of CORDIC architecture

The hardware implementation for CORDIC arithmetic is shown in Figure.2. It requires three registers for x, y and z, a look up table to store the values of tan $^{-1}$ 2 $^{-i}$ and two shifter to supply the terms 2^{-i} x and 2^{-i} y to the adder/sub tractor units. All multiplication operations are reduced to simple shift operations. Form Figure.6.the branch consists of an adder-sub tractor combination, a shift unit and a register for buffering the output. At the beginning of a calculation initial values are fed into the register by the multiplexer where the MSB of the stored value in the z-branch determines the operation mode for the adder-sub tractor. Signals in the x and y branch pass the shift units and are then added to or subtracted from the unshifted signal in the opposite path.

5. RESULTS AND DISCUSSIONS

Phase Accumulator is a heart of the DDS. The phase accumulator can be implemented in Pipelined Carry Look Ahead architecture to achieve high throughput. A 32-bit phase accumulator has been chosen to be implemented in this work. The well-known Carry-look ahead Adder (CLA) technique has been used to implement the phase accumulator. The phase accumulator block is a 32 bit accumulator that is pipelined in 8 bit sections. It has been implemented using VHDL and simulated in XilinxISE9.2i. This module consist of frequency register, 32 bit hybrid wave pipelined adder and phase register. With every clock pulse the contents of the frequency register (FCW) is added to that of PA.

The frequency of the output wave depends on the overflow rate of the PA and the frequency tuning word. This overflow rate depends on the frequency tuning word stored in the phase register. The 32 bit adder output is truncated into 14 bits [12] only higher order bits are taken. The phase value will not be affected if the truncation is carried out because the phase information is only available in higher order bits. This module produces output which is 12bit wide and these outputs are input of the pipelined CORDIC module. In conventional DDS, ROM Look up Table method is used for Phase to Amplitude Conversion. This ROMLUT method occupies maximum memory space and it utilizes more number of resources on FPGAs. In order minimize the area utilization in FPGA, the pipelined CORDIC architecture is used in this design instead of ROMLUT.

To generate an output frequency of 10 MHz with a reference clock frequency of 50MHz, a frequency tuning word (*M*) of 33333333H is stored in the frequency Register. The value of the frequency tuning word (*M*) is calculated using the frequency tuning equation. The Phase Accumulator is 32-bits wide. This control word *M* is added to the previous value of PA with each clock pulse. The output of the Phase accumulator is increased linearly. This linear output must be converted to sinusoidal.

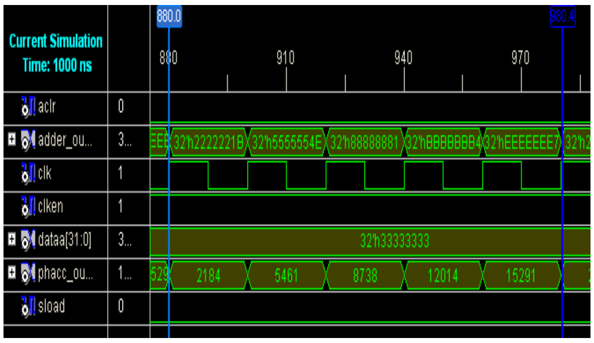


Figure.3. Phase accumulator output for fout = 10MHz and FCW=33333333H

Figure.3. shows the simulation result of proposed DDS phase accumulator with 10MHz output frequency. The 32 bit adder output is truncated into 14 bits [12] [13], only higher order bits are taken. The phase value will not be affected if the truncation is carried out because the phase information is only available in higher order bits. This module produces phase register output which is 13bit wide along with sine and cosine output. Figure.4. Shows the simulation result of pipelined CORDIC architecture. Xilinx ISE simulation tool produces the output in form of digital values. Using this CORDIC output values the sine and Cosine waveform are plotted and shown in Figure.5.

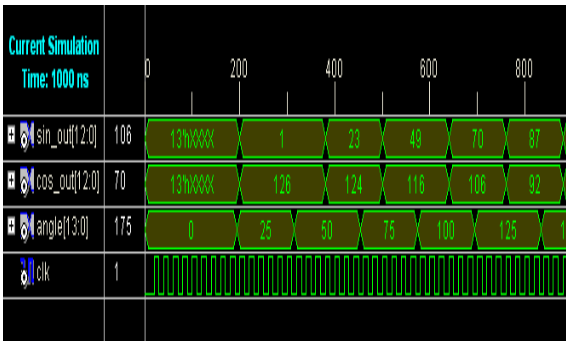
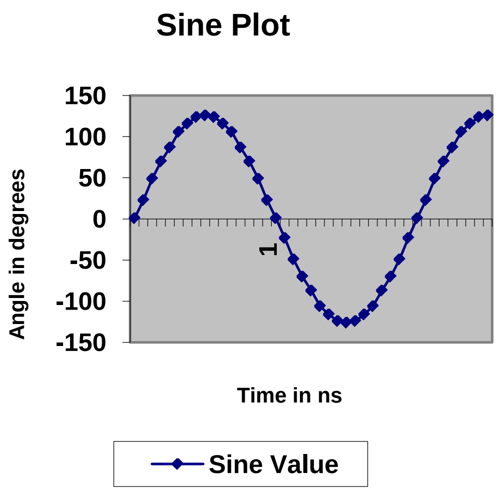


Figure.4.Sine and Cosine output of CORDIC Module

Volume 2 No.6, June 2012 ISSN 2224-3577
International Journal of Science and Technology
©2012 IJST. All rights reserved
http://www.ejournalofsciences.org



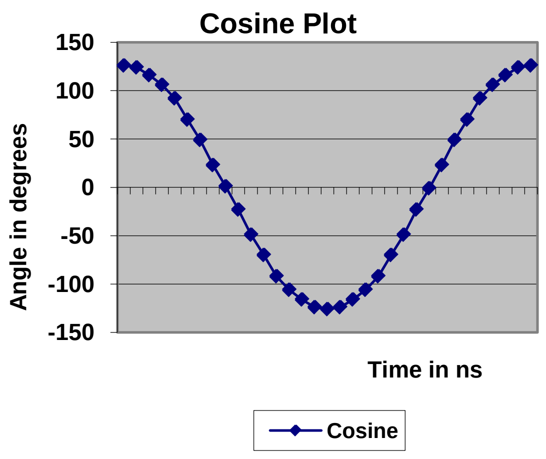


Figure.5: Sine and Cosine Sample Values Plotted for pipelined CORDIC Architecture

The output frequency of the DDFS is directly proportional to the frequency tuning word. Therefore, larger the frequency tuning word, higher is the output frequency and faster the PA overflows. In order to increase the high throughput and overall performance, the pipelined CORDIC algorithm is used. VHDL code is developed for proposed CORDIC architecture and simulated in XILINX ISE9.2 tools. This simulation, performed with a reference clock of 50MHz. The design summary of hybrid wave pipelined with CORDIC DDS using Spartan III FPGA is shown in Figure.6. The frequency resolution is 0.0116Hz and the SFDR is calculated to be 112dB.

This section presents simulation results of our proposed DDS architecture. Simulations are done with input bits of 32 bits and its frequency resolution 0.01164Hz and Phase resolution is 0.022°. The input of CORDIC module is 14bits and this block produces two outputs such as sine and cosine waveform. Synthesis results are summarized in Table.1. The number of slices, Look up Tables (LUTs) and registers are calculated for two designs: ROMLUT DDS and Pipelined CORDIC DDS. These results point out the area and speed optimization of the proposed method. In fact, the proposed method uses 135 slice elements and 194 LUTs against 165 slices and 226 LUTs for the ROMLUT method. Compare to other technique the speed is also increased and its maximum frequency is 184.76MHz. The proposed method is simulated and implemented on Spartan3 Xilinx FPGA. At the initial stage of the proposed method consists of pipelined CLA phase accumulator which is used to increase the speed.

Table 1: Speed and Area Performance

Technique	No of Slices used	No of 4 input LUTs	Number of Register	Maximum frequency (MHz)
ROMLUT DDS	165	226	245	140.73
Pipelined CORDIC DDS	135	194	175	184.76

The second stage of the proposed method uses CORDIC algorithm instead of Sine ROMLUT. CORDIC is more suitable for FPGA implementation which increases the speed and reduces the number of slices, registers and LUTs. To generate an output frequency of 10 MHz with a reference clock frequency of 50MHz, a frequency tuning word (*M*) of 33333333H is stored in the frequency Register. The value of the frequency tuning word (*M*) is calculated using the frequency tuning equation. The Phase Accumulator is 32-bits wide. This control word *M* is added to the previous value of PA with each clock pulse. The output of the Phase accumulator is increased linearly. This linear output must be converted to sinusoidal.

6. CONCLUSION

This proposed method takes advantage of Pipelined CLA phase accumulator scheme with CORDIC which increases throughput and decreases slices, registers and LUTS in FPGA. This proposed DDS uses a large phase accumulator to perform a high frequency resolution. The architecture has been compared with the prior art architectures in simulations and measurements. The simulated and measured results demonstrated that the maximum frequency is 184.76MHz, frequency resolution of 0.0023Hz and Phase resolution is 0.088 degree. The spurious performance is improved at 112dB. This proposed design provides better SFDR, high maximum frequency; High throughput and minimum resource utilization compare to conventional DDS architectures.

REFERENCES

[1] Jeffrey H Reed,” Software Radio: A Modern Approach to Radio Engineering, Prentice Hall,2002

[2] Jouko Vankka,” Direct Digital Synthesizers: Theory, Design and Applications”, London, Kluwer Academic Publishers, 2001.

[3] A Technical Tutorial on Digital Signal Synthesis, Online Available WWW: http://www.analog.com/UploadedFiles/Tutorials/450968421DDS_Tutorial_rev12-2-99.

[4] D.R. Moran, J.G. Menoyo and J.L. Martin, “Digital Frequency Synthesizer Based on Two Coprime Moduli DDS”. IEEE Transaction on Circuit and Systems II, vol. 53, no.12, pp. 1388—1392, 2006.

[5] Lakshmi S. Jyothi Chimakurthy, Malinky Ghosh, Fa Foster Dai, and Richard C. Jaeger, “A Novel DDS Using Nonlinear ROM Addressing With Improved Compression Ratio and Quantization Noise” IEEE Transactions on Ultrasonic, Ferroelectrics, and Frequency control, vol. 53, no. 2, February 2006.

[6] A.L.MacEwan and .S.Collin “ Efficient , ROMLess DDFS using non-linear Interpolation and non- linear DAC” Analog Integrated Circuit Signal Processing, Springer, Vol. 48, pp231-237, 2006

[7] D. Timmermann, H. Hahn, B. J. Hosticka, and G. Schmidt, “A programmable CORDIC chip for digital signal processing applications,” *IEEE Journal of Solid-State Circuits*, vol. 26, no. 9, pp. 1317–1321, 1991.

[8] J. E. Volder, “The CORDIC trigonometric computing technique,” *IRE Transactions on Electronic Computers*, vol. 8, no. 3, pp. 330–334, 1959.

[9] J. E. Volder, “The birth of CORDIC,” *Journal of VLSI Signal Processing*, vol. 25, no. 2, pp. 101–105, 2000.

[10] J. S. Walther, “A unified algorithm for elementary functions,” in *Proceedings of the AFIPS Spring Joint Computer Conference*, pp. 379–385, May 1971.

[11] J. S.Walther, “The story of Unified CORDIC,” *Journal of VLSI Signal Processing*, vol. 25, no. 2, pp. 107–112, 2000.

[12] D. E. Metafas and C. E. Goutis, “A DSP processor with a powerful set of elementary arithmetic operations based on cordic and CCM algorithms,” *Microprocessing and Microprogramming*, vol. 30, no. 1–5, pp. 51–57, 1990.

[13] Chang Yong Kang and Earl E. Swartzlander,” Digit-Pipelined Direct Digital Frequency Synthesis Based on Differential CORDIC” IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 53, NO. 5, MAY 2006

[14] H.Wang, P. Leray, and J. Palicot, “Reconfigurable architecture for MIMO systems based on CORDIC operators,” *Comptes Rendus Physique*, vol. 7, no. 7, pp. 735–750, 2006.

[15] A. Meyer-Base, R. Watzel, U. Meyer-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, 2003.

[16] Maher Jridi**,** Ayman Alfalou “Direct Digital Frequency Synthesizer with CORDIC Algorithm and Taylor Series Approximation for Digital Receivers” European Journal of Scientific Research, ISSN 1450-216X Vol.30 No.4 (2009), pp.542-553

[17] Tze-Yun Sung, Hsi-Chin Hsin, Lu-Ting Ko “High-SFDR and Multiplierless Direct Digital Frequency Synthesizer” WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, Issue 6, Volume 8, June 2009.

[18] Eugene Grayver, Babak Daneshrad,” DIRECT DIGITAL FREQUENCY SYNTHESIS USING A MODIFIED CORDIC”, IEEE, 1998.

[19] Davide De Caro, Nicola Petra, and Antonio G. M. Strollo, ” Digital Synthesizer/Mixer With Hybrid CORDIC–Multiplier Architecture: Error Analysis and Optimization,” IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 56, NO. 2, FEBRUARY 2009.

[20] Oskar Mencer, Martin Morf” Parallel, Pipelined CORDICs for Reconfigurable Computing”, DARPA Grant Nr. DABT63-96-C-0106.

[21] Chen shijie , Houjun Wang,” A Study bf Signal Generation Based on CORDIC Algorithm,” IEEE ex plore,2005.

[22] R. Andraka, “A survey of CORDIC algorithms for FPGA based computers,” in Proceedings of the 6th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA ’98), pp. 191–200, February 1998.

[23] M. Kuhlmann and K. K. Parhi, “P-CORDIC: a pre-computation based rotation CORDIC algorithm,”

EURASIP Journal on Applied Signal Processing, vol. 2002, no. 9, pp. 936–943, 2002.

[24] M. Kuhlmann and K. K. Parhi, "A high-speed CORDIC algorithm and architecture for DSP applications," in Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS '99), pp. 732–741, October 1999.

[25] 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, no. 9, pp. 989–995, 1991.

[26] D. Timmermann, H. Hahn, and B. J. Hosticka, "Low latency time CORDIC algorithms," IEEE Transactions on Computers, vol. 41, no. 8, pp. 1010–1015, 1992.

[27] B. Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," Microelectronics Journal, vol. 33, no. 1-2, pp. 77–89, 2002.

[28] T.-B. Juang, S.-F. Hsiao and M.-Y. Tsai, "Para-CORDIC: parallel CORDIC rotation algorithm," IEEE Transactions on[31] Circuits and Systems I, vol. 51, no. 8, pp. 1515–1524, 2004.

[29] H. S. Kebbati, J. Ph. Blonde, and F. Braun, "A new semi-flat architecture for high speed and reduced area CORDIC chip," Microelectronics Journal, vol. 37, no. 2, pp. 181–187, 2006.

[30] T. Srikanthan and B. Gisuthan, "A novel technique for eliminating iterative based computation of polarity of microrotations in CORDIC based sine-cosine generators," Microprocessors and Microsystems, vol. 26, no. 5, pp. 243–252, 2002.

[31] J. Duprat and J.-M. Muller, "The CORDIC algorithm: new results for fast VLSI implementation," IEEE Transactions on Computers, vol. 42, no. 2, pp. 168–178, 1993.

[32] D. S. Phatak, "Double step branching CORDIC: a new algorithm for fast sine and cosine generation," IEEE Transactions on Computers, vol. 47, no. 5, pp. 587–602, 1998

[33] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley & Sons, New York, NY, USA, 1999.

[34] P. W. Baker, "Suggestion for a fast binary sine/cosine generator," IEEE Transactions on Computers, vol. 25, no. 11, pp. 1134– 1136, 1976.

[35] Y. H. Hu, "Pipelined CORDIC architecture for the implementation of rotational based algorithm," in Proceedings of the International Symposium on VLSI Technology, Systems and Applications, p. 259, May 1985.