CORDIC Based Direct Digital Frequency Synthesizer

Thesis submitted in the partial fulfillment of requirement for the award of degree of

Master of Technology

in

VLSI Design & CAD

Submitted by:

Amit Goyal

Roll No : 601061004

Under the guidance of:

Mr. H.K.S Randhawa

Assistant Professor

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA – 147004 (PUNJAB)

June 2012

DECLARATION

I, **Amit Goyal**, hereby certify that the work which is being presented in this thesis entitled "**CORDIC BASED DIRECT DIGITAL FREQUENCY SYNTHESIZER**" by me in partial fulfillment of the requirements for the award of degree of Master of Technology in VLSI Design from Thapar University (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of **Mr. H.K.S.Randhawa**.

The matter presented in this thesis has not been submitted in any other University / Institute for the award of any other degree.

Date: 29/6/12

Amit Goyal
Roll No. 601061004

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

29/6/2K12
Date:

Mr. H.K.S.Randhawa
Assistant Professor
ECED

Countersigned by:

Dr. Rajesh Khanna
Professor and Head ECED
Thapar University, Patiala
Date:

Dr. S.K. Mohapatra
Dean of Academic Affairs
Thapar University Patiala
Date:

ACKNOWLEDGEMENT

To discover, analyze and to present something new is to venture on an untrodden path towards and unexplored destination is an arduous adventure unless one gets a true torchbearer to show the way. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various people. Words are often too less to reveals one's deep regards. I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis. I acknowledge with gratitude and humility my indebtedness to **Mr. H.K.S Randhawa, Assistant Professor,** Electronics and Communication Engineering Department, Thapar University, Patiala, under whose guidance I had the privilege to complete this thesis. I wish to express my deep gratitude towards her for providing individual guidance and support throughout the thesis work.

I convey my sincere thanks to **Head of the Department, Dr. Rajesh Khanna** as well as **PG Coordinator, Dr. Kulbir Singh, Assistant Professor,** Electronics and Communication Engineering Department, entire faculty and staff of Electronics and Communication Engineering Department for their encouragement and cooperation.

My greatest thanks are to all who wished me success especially my parents. Above all I render my gratitude to the Almighty who bestowed self-confidence, ability and strength in me to complete this work for not letting me down at the time of crisis and showing me the silver lining in the dark clouds. I do not find enough words with which I can express my feelings of thanks to my dear friends for their help, inspiration and moral support which went a long way in successful competition of the present study.

Abstract

Traditional designs of high bandwidth frequency synthesizers employ the use of a phase-locked-loop (PLL). A direct digital frequency synthesizer (DDFS) provides many significant advantages over the PLL approaches. Fast settling time, sub-Hertz frequency resolution, continuous-phase switching response and low phase noise are features easily obtainable in the DDFS systems. Although the principle of the DDFS has been known for many years, the DDFS did not play a dominant role in wideband frequency generation until recent years. Earlier DDFSs were limited to produce narrow bands of closely spaced frequencies, due to limitations of digital logic and D/A-converter technologies. Recent advantages in integrated circuit (IC) technologies have brought about remarkable progress in this area. By programming the DDFS, adaptive channel bandwidths, modulation formats, frequency hopping and data rates are easily achieved. This is an important step towards a "software-radio" which can be used in various systems. The DDFS could be applied in the modulator or demodulator in the communication systems. The applications of DDFS are restricted to the modulator in the base station. **D**irect digital frequency synthesizer (DDFS) is a digital technique for generating sinusoids. Unlike conventional analog oscillator structures, DDFS can be applied where fast frequency switching, fine tuning and a coherent phase relationship among sinusoids are required. The circular-mode CORDIC (coordinate rotation digital computer) algorithm is an iterative method to compute sine/cosine values, in which an initial vector is rotated by a predetermined sequence of sub-angles in such a way that the summation of the rotations approaches the given angle. CORDIC has been a widely-adopted method for implementing the sine/cosine generator in DDFS. When compared to the ROM look-up-table approach, where all the required sine/cosine sample values are stored in a ROM, the CORDIC approach does not lead to the exponential growth of the hardware. Recent advances in IC fabrication technology, particularly CMOS, coupled with advanced DSP algorithms and architectures are providing possible single-chip DDFS solutions to complex communication and signal processing subsystems as modulators, demodulators, local oscillators (LOs), and programmable clock generators. The DDFS addresses a variety of applications, including cable modems, measurement equipments, arbitrary waveform generators,

cellular base stations and wireless local loop base stations.

The aim of this report is to compare some DDFS architecture and to try to find an optimized DDFS architecture .In this we also study about CORDIC (Coordinate rotation digital computer).Because CORDIC is main part of DDFS.

List of figures

List of tables

Contents

Chapter 1

INTRODUCTION

1.1 OVERVIEW

A major advantage of a direct digital frequency synthesizer (DDFS) is that its output frequency, phase and amplitude can be precisely and rapidly manipulated under digital processor control. Other inherent DDFS attributes include the ability to tune with extremely fine frequency and phase resolution, and to rapidly "hop" between frequencies. These combined characteristics have made the technology popular in military radar and communications systems. In fact, DDFS technology was previously applied almost exclusively to high-end and military applications: it was costly, power-hungry, difficult to implement, and required a discrete high speed D/A converter. It is easy to include different modulation capabilities in the DDFS by using digital signal processing methods, because the signal is in digital form [1]. By programming the DDFS, adaptive channel bandwidths, modulation formats, frequency hopping and data rates are easily achieved. The flexibility of the DDFS makes it ideal for signal generator for software radio. The digital circuits used to implement signal-processing functions do not suffer the effects of thermal drift, aging and component variations associated with their analog counterparts. The implementation of digital functional blocks makes it possible to achieve a high degree of system integration. Recent advances in IC fabrication technology, particularly CMOS, coupled with advanced DSP algorithms and architectures are providing possible single-chip DDS solutions to complex communication and signal processing subsystems as modulators, demodulators, local oscillators (LOs), and programmable clock generators. The DDS addresses a variety of applications, including cable modems, measurement equipments, arbitrary waveform generators, cellular base stations and wireless local loop base stations.

1.2 DDFS AND CORDIC

DIRECT digital frequency synthesizer (DDFS) is a digital technique for generating sinusoids. Unlike conventional analog oscillator structures, DDFS can be applied where fast frequency switching, fine tuning and a coherent phase relationship among sinusoids are required [2].

Direct digital synthesis (DDS) is a technique for using digital data processing blocks as a means to generate a frequency- and phase-tunable output signal referenced to a fixed frequency precision clock source. In essence, the reference clock frequency is "divided down" in a DDS architecture by the scaling factor set forth in a programmable binary tuning word. The tuning word is typically 24-48 bits long which enables a DDS implementation to provide superior output frequency tuning resolution.

Today's cost-competitive, high-performance, functionally-integrated, and small package-sized DDS products are fast becoming an alternative to traditional frequency-agile analog synthesizer solutions. The integration of a high-speed, high-performance, D/A converter and DDS architecture onto a single chip (forming what is commonly known as a Complete-DDS solution) enabled this technology to target a wider range of applications and provide, in many cases, an attractive alternative to analog-based PLL synthesizers. For many applications, the DDS solution holds some distinct advantages over the equivalent agile analog frequency synthesizer employing PLL circuitry. There are so many advantages of DDFS. Among those some are:

- Micro-Hertz tuning resolution of the output frequency and sub-degree phase tuning Capability, all under complete digital control.
- Extremely fast "hopping speed" in tuning output frequency (or phase), phase-continuous frequency hops with no over/undershoot or analog-related loop settling time anomalies.

The circular-mode CORDIC (coordinate rotation digital computer) algorithm is an iterative method to compute sine/cosine values, in which an initial vector is rotated by a predetermined sequence of sub-angles in such a way that the summation of the rotations approaches the given angle [3]. CORDIC has been a widely-adopted method for implementing the sine/cosine generator in DDFS [4]-[5]. When compared to the ROM look-up-table approach, where all the required sine/cosine sample values are stored in a ROM [5], the CORDIC approach does not lead to the exponential growth of the hardware.

CORDIC is a very interesting technique for phase to sine amplitude conversion. This algorithm proposed by J.E.Volder utilizes dynamic transformation rather than ROM static addressing [2]. 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. The trigonometric CORDIC algorithms were originally developed as a digital solution for real-time navigation problems. The original work is credited to Jack Volder. Extensions to the CORDIC theory based on work by John Walther and others provide solutions to a broader class of functions. The CORDIC algorithm has found its way into diverse applications including the microprocessors, calculator, radar signal processors [6] and robotics. CORDIC rotation has also been proposed for computing Discrete Fourier [7], Discrete Cosine, Discrete Hartley, and filtering, and solving linear systems [8].
Now a days world of information interchange revolves around transmission and viewing real-time images. Many of the digital signal processing (DSP) applications try to closely simulate real life images, Speed, Clarity and resemblance to real time objects are some of the many issues to be addressed in order to achieve this goal. Trigonometric functions calculation is one of the primary tasks performed in DSP applications. For a long time microprocessor-based systems have been used to perform this task. Software algorithms used by the processors do not meet the highly demanding needs of all DSP tasks.

Using hardware systems to perform these DSP tasks is a competent solution to this problem. FPGA are often used as coprocessors to perform all the high speed tasks that cannot be achieved by microprocessors. FPGAs are chosen because they are on site programmable and are highly suitable for hardware implementations. The software solutions adapted by the microprocessors to implement trigonometric functions are computer intensive. They do not suit hardware platforms because they need complex circuits to perform the mathematical operations. Hence hardware algorithms are adapted for the calculation of trigonometric functions. To calculate trigonometric functions one such algorithm is the CORDIC algorithm. It was developed by J.E.Volder. His objective was to build a real time navigational computer for use on aircrafts [9], so he was primarily interested in computing trigonometric functions. Due to simplicity of the involved operations in the CORDIC algorithm is very well suited for

VLSI implementations. Compared to other approaches, Cordic is a clear winner when a hardware multiplier is unavailable.

1.3 LITERATURE SURVEY

The very earliest history of the CORDIC computing technique—a highly efficient method to compute elementary functions is presented. The CORDIC technique was born out of necessity, the incentive being the replacement of the analog navigation computer of the B-58 aircraft by a high accuracy, high-performance digital computer. The revolutionary development of the CORDIC technique is presented, along with details of the very first implementations: the CORDIC I prototype and the CORDIC II airborne digital navigation computer.

In early 1956 the aeroelectronics department of Convair, Fort Worth, was given the task of determining the feasibility of replacing the analog computer-driven navigation system of the B-58 bomber (see Fig. 1) with a digital computer. This replacement effortwas deemed necessary because of the limited accuracy of analog computing elements.

At that time, digitalization of the B-58 navigation system was considered a formidable task without any assurance of worthwhile results. Transistors were new and limited to a 250 KHz logic rate. The main challenge was the real-time calculation of the complicated navigation equations required for determining present position on a spherical earth.

By then, digital differential analyzers had been developed that were capable of efficiently solving continuous navigation problems, but they could not produce solutions in real-time during flights near the North Pole. Also, they were too slow in providing solutions for the discontinuous problems of fix-taking from either startracking or radar ground sightings. Therefore, an entireword transfer type of computation was definitely necessary. However, the existing trigonometric algorithms necessary for navigation were too time-consuming for the real-time requirements of the B-58.

Most navigation system specialists agreed that the existing B-58 navigation computer was an ingenuous device utilizing analog resolvers to compute, in real time, the complex trigonometric relationships necessary for navigation over a spherical earth. Each resolver was capable of performing either a rotation of input coordinates or inversely determining the magnitude and angle of the vector defined by the input coordinates also called vectoring .

Trying to solve navigation problems without resolver capabilities is an extremely complicated problem. Resolver capability allows solution flow diagrams to be drawn with interconnected resolvers as was shown in the original CORDIC paper. For example, in great

circle navigation, the solution of course angle and distance to destination requires a network of only 5 interconnected resolvers.

Finally, in late 1958, permission was obtained from both Convair and the Air Force to present a technical description of CORDIC at the March 1959 Western Joint Computer Conference. Also, permission was obtained to build a demonstrational limited capability navigation system using the new CORDIC computing concept. This system, identified as CORDIC I, was completed in early 1960. CORDIC I only solved the problem of fix-taking from inputs of a simulated star-tracker. Although only a laboratory prototype operating at a clock rate of 96.5773 KHz, it created significant interest with Air Force, and management persons concerned with navigation. As a result of the CORDIC I demonstration, Convair was given authorization to develop a flyable demonstration model that could solve the radar fix-taking problem. This new model, identified as CORDIC II, was highly advanced for its time. Its performance was estimated at a factor of seven times that of contemporary general purpose computers, mainly thanks to the revolutionary development of the CORDIC algorithm.

Operating at a clock frequency of 198.4 KHz, The CORDIC II was fully transistorized, using diode transistor logic (DTL) circuits. It had a high-density assembly of miniature printed circuit boards, each containing in the order of 10 gates (multi-input ANDs, ORs, invertors). For storage, it employed a magnetic drum with multiple read/write heads, and rotating at 6000 rpm. The CORDIC II was highly versatile, and could be applied to a wide variety of navigation and control applications. It had an instruction set of 31 operations including rotation, vectoring, binary to BCD conversion, multiplication, division, addition and subtraction. It was highly reliable, capable of maintaining fault-free operation under the severe conditions (vibration, pressure, temperature) of the environment aboard the B-58 aircraft. It was even capable of withstanding the shock of explosive decompression and crash landings. It successfully passed the airborne radar fix-taking test in early 1962.

Before completion of CORDIC I, the editors of the IRE Transaction on Electronic Computers requested permission to reprint my WJCC paper in their September 1959 issue. It was published back-to-back with Dan Daggett's publication on using CORDIC for the conversion between binary to BCD representation [4]. Industry acceptance of these technical papers was immediate, especially by those involved in military navigation computers. Versions of CORDIC were soon incorporated into the navigation computers built by Martin-Orlando, Computer Control, Litton, Kearfott, Lear-Siegler, Sperry, Raytheon, Collins Radio, and later by Univac and Hughes. In 1966 Hewlett Packard developed a decimal version that computed trigonometric and hyperbolic solutions in their scientific calculators. However, in spite of the

success of the CORDIC solution for digitizing the B-58 navigation system, the Air Force terminated construction of B-58 bombers in favour of support for the development of ballistic missile systems. Out of the 117 B-58's that were ever built, only 5 survived to present day at various aircraft musea, the remainder being scrapped in the 1970's. The CORDIC approach continued to find use in other navigation systems, Loran C systems and numerical processors for general purpose computers.

The Coordinate Rotation Digital Computer (CORDIC) was introduced in 1959 by Volder.

It is an easy-to-implement and versatile algorithm widely used for digital signal processing applications. It computes iteratively the rotation of a two-dimensional vector using only add and shift operations. CORDIC has been traditionally used for hardware implementations. In several algorithms which admit efficient implementation using CORDIC were reviewed: linear transforms, digital filtering, and matrix based DSP computing algorithms. It was shown that CORDIC-based architectures are a very appealing alternative to conventional multiply-and-add hardware. However, CORDIC may be also applied to implement different communication subsystems found in a digital radio: direct digital synthesizers; amplitude modulation (AM), phase modulation (PM), and frequency modulation (FM) analog modulators; amplitude shift keying (ASK), phase shift keying (PSK), and frequency shift keying (FSK) modulators, up-down converters of in-phase and quadrature signals, full mixers for complex signals, and phase detection for synchronizers. First, CORDIC is introduced as a computational resource that is able to rotate a vector by an angle, and convert from Cartesian to polar coordinates; also shown is a generic scheme suitable for implementing the different tasks required in communication systems. Second, several applications are commented on: direct digital synthesis; frequency, phase, and amplitude modulation; up-down conversion; and frequency and phase synchronization for quadrature amplitude modulation (QAM) and orthogonal frequency-division multiplexing (OFDM) systems. Third, the CORDIC algorithm is exposed, and some implementation issues to improve its performance are included: how to enhance the spurious free dynamic range (SFDR) in direct digital synthesizers (DDS); how to avoid the pi/2 multiplier of the phase accumulator; and how to simplify the computation of the scaling factor.

For an easy understanding of how to use the CORDIC algorithm in the implementation of digital intermediate frequency (IF) communications systems, CORDIC is presented in this section only as a computational resource with three inputs (X0, Y0, and Z0) and three outputs (XN, YN, and ZN) that allows performing the following operations :

• Rotation of a vector (I,Q) by an angle q when it is operating in rotation mode (RM); the rotated output vector is multiplied by a constant value K

• Cartesian-to-polar conversion, when it is operating in vectoring mode (VM); themodulus of the vector is also scaled by K

The generation of many different frequencies from a single stable source frequency is commonly achieved with analog circuits, or combinations of analog and digital circuits. All of these approaches generate a large set of frequencies from a single source in the analog or continuous sense by division, phase lock, mixing, or a combination of such techniques. An attractive alternate approach is the generation of a set of sampled sinusoids by digital computation of some kind (including sine table lookup) at the sampling times.

Direct digital synthesis (DDS) is a technique for using digital data processing blocks as a means to generate a frequency and phase-tunable output signal referenced to a fixedfrequency precision clock source. In essence, the reference clock frequency is 'divided down' in DDS architecture by the scaling factor set forth in a programmable binary tuning word. The tuning word is typically 24-48 bits long which enables a DDS implementation to provide superior output frequency tuning resolution. A direct digital synthesizer (DDS) provides many significant advantages over traditional approaches :

- Micro-Hertz tuning resolution of the output frequency and sub-degree phase tuning capability, all under complete digital control.
- Extremely fast "hopping speed" in tuning output frequency (or phase), phase-continuous frequency hops with no over/undershoot or analog-related loop settling time anomalies.
- The DDS digital architecture eliminates the need for the manual system tuning and tweaking associated with component aging and temperature drift in analog synthesizer solutions.
- The digital control interface of the DDS architecture facilitates an environment where systems can be remotely controlled and minutely optimized under processor control.
- When utilized as a quadrature synthesizer, DDS afford unparalleled matching and control of I and Q synthesized outputs.

A major advantage of a direct digital synthesizer (DDS) is that its output frequency, phase and amplitude can be precisely and rapidly manipulated under digital processor control. Other inherent DDS attributes include the ability to tune with extremely fine frequency and phase resolution, and to rapidly "hop" between frequencies. These combined characteristics

have made the technology popular in military radar and communications systems. In fact, DDS technology was previously applied almost exclusively to high-end and military applications: it was costly, power-hungry, difficult to implement, and required a discrete high speed D/A converter. Due to improved integrated circuit (IC) technologies, they now present a viable alternative to analog based phase-locked loop (PLL) technology for generating agile analog output frequency in consumer synthesizer applications. It is easy to include different modulation capabilities in the DDS by using digital signal processing methods, because the signal is in digital form. By programming the DDS, adaptive channel bandwidths, modulation formats, frequency hopping and data rates are easily achieved. The flexibility of the DDS makes it ideal for signal generator for software radio. The digital circuits used to implement signal-processing functions do not suffer the effects of thermal drift, aging and component variations associated with their analog counterparts. The implementation of digital functional blocks makes it possible to achieve a high degree of system integration. Recent advances in IC fabrication technology, particularly CMOS, coupled with advanced DSP algorithms and architectures are providing possible single-chip DDS solutions to complex communication and signal processing subsystems as modulators, demodulators, local oscillators (LOs), programmable clock generators, and chirp generators. The DDS addresses a variety of applications, including cable modems, measurement equipments, arbitrary waveform generators,cellular base stations and wireless local loop base stations. The DDS is better suited to base stations than to mobiles, because power consumption is high in the wide output bandwidth DDS. The scaling of the IC technologies constantly reduces power consumption in digital circuitry. The same benefit is, however, not easily achieved in analog circuits. Therefore, the DDS might also be suitable for the mobiles in the future. The applications of DDS are restricted to the modulator in the base station. It follows that spurs and noise are the main concern, not power consumption. The spurious performance is not good in the wide output bandwidth DDS because of analog errors in D/A conversion.

The analog part of the DDS includes a D/A converter and a low pass filter. The DDS is a mixed signal device where sensitive analog blocks must tolerate the distortion from digital rail-to-rail signals. The cross-talk issue must be focused when the integration level increases. Another problem is the limited speed and resolution in D/A conversion. Unfortunately, the development of D/A converters does not keep up with the capabilities of digital signal processing with faster technologies.

Traditional designs of high bandwidth frequency synthesizers employ the use of a phaselocked- loop (PLL). A direct digital synthesizer (DDS) provides many significant

advantages over the PLL approaches. Fast settling time, sub-Hertz frequency resolution, continuous-phase switching response and low phase noise are features easily obtainable in the DDS systems. Although the principle of the DDS has been known for many years, the DDS did not play a dominant role in wideband frequency generation until recent years. Earlier DDSs were limited to produce narrow bands of closely spaced frequencies, due to limitations of digital logic and D/A converter technologies. Recent advantages in integrated circuit (IC) technologies have brought about remarkable progress in this area. By programming the DDS, adaptive channel bandwidths, modulation formats, frequency hopping and data rates are easily achieved. This is an important step towards a "software-radio" which can be used in various systems.

Chris K Cockrum shows that the CORDIC (COordinate Rotation by DIgital Computer) algorithm gives significant efficiency gains over a Taylor approximation for calculating the sine and cosine functions to a given precision for many applications implemented in hardware or a microcontroller. In the case where the processor has no dedicated multiplier or trigonometric algorithm hardware, it is shown that the CORDIC is over 14 times more efficient when calculating the sine and cosine functions to 10 decimal digits of precision. This implementation is tailored towards use by Amateur Radio enthusiasts in the digital down-converter (DDC) of software defined radio (SDR) applications on dedicated hardware.

Direct digital synthesis (DDS) is a method of producing an analog waveform—usually a sine wave by generating a time-varying signal in digital form and then performing a digital-to-analog conversion. Because operations within a DDS device are primarily digital, it can offer fast switching between output frequencies, fine frequency resolution, and operation over a broad spectrum of frequencies. With advances in design and process technology, today's DDS devices are very compact and draw little power.

The ability to accurately produce and control waveforms of various frequencies and profiles has become a key requirement common to a number of industries. Whether providing agile sources of low-phase-noise variable-frequencies with good spurious performance for communications, or simply generating a frequency stimulus in industrial or biomedical test equipment applications, convenience, compactness, and low cost are important design considerations.

Many possibilities for frequency generation are open to a designer, ranging from phase-locked-loop (PLL)-based techniques for very high-frequency synthesis, to dynamic programming of digital-to-analog converter (DAC) outputs to generate arbitrary waveforms at lower frequencies. But the DDS technique is rapidly gaining acceptance for solving

frequency (or waveform) generation requirements in both communications and industrial applications because single-chip IC devices can generate programmable analog output waveforms simply and with high resolution and accuracy.

Furthermore, the continual improvements in both process technology and design have resulted in cost and power consumption levels that were previously unthinkably low. For example, the AD9833, a DDS-based programmable waveform generator, operating at 5.5 V with a 25-MHz clock, consumes a maximum power of 30 milliwatts.

DDS devices like the AD9833 are programmed through a high speed serial peripheral-interface (SPI), and need only an external clock to generate simple sine waves. DDS devices are now available that can generate frequencies from less than 1 Hz up to 400 MHz (based on a 1-GHz clock). The benefits of their low power, low cost, and single small package, combined with their inherent excellent performance and the ability to digitally program (and re-program) the output waveform, make DDS devices an extremely attractive solution preferable to less-flexible solutions comprising aggregations of discrete elements.

Applications currently using DDS-based waveform generation fall into two principal categories: Designers of communications systems requiring agile (i.e., immediately responding) frequency sources with excellent phase noise and low spurious performance often choose DDS for its combination of spectral performance and frequency-tuning resolution. Such applications include using a DDS for modulation, as a reference for a PLL to enhance overall frequency tunability, as a local oscillator (LO), or even for direct RF transmission.

Alternatively, many industrial and biomedical applications use a DDS as a programmable waveform generator. Because a DDS is digitally programmable, the phase and frequency of a waveform can be easily adjusted without the need to change the external components that would normally need to be changed when using traditional analog programmed waveform generators. DDS permits simple adjustments of frequency in real time to locate resonant frequencies or compensate for temperature drift. Such applications include using a DDS in adjustable frequency sources to measure impedance (for example in an impedance-based sensor), to generate pulse-wave modulated signals for micro-actuation, or to examine attenuation in LANs or telephone cables.

The key performance specifications of a DDS based system are Phase noise, jitter, and spurious-free dynamic range (SFDR).Phase noise is a measure (dBc/Hz) of the short-term frequency instability of the oscillator. It is measured as the single-sideband noise resulting from changes in frequency (in decibels below the amplitude at the operating frequency of the

oscillator using a 1-Hz bandwidth) at two or more frequency displacements from the operating frequency of the oscillator. This measurement has particular application to performance in the analog communications industry.

Jitter is the dynamic displacement of digital signal edges from their long-term average positions, measured in degrees rms. A perfect oscillator would have rising and falling edges occurring at precisely regular moments in time and would never vary. This, of course, is impossible, as even the best oscillators are constructed from real components with sources of noise and other imperfections. A high-quality, low-phase-noise crystal oscillator will have jitter of less than 35 picoseconds (ps) of period jitter, accumulated over many millions of clock edges.

Jitter in oscillators is caused by thermal noise, instabilities in the oscillator electronics, external interference through the power rails, ground, and even the output connections. Other influences include external magnetic or electric fields, such as RF interference from nearby transmitters, which can contribute jitter affecting the oscillator's output. Even a simple amplifier, inverter, or buffer will contribute jitter to a signal.

Thus the output of a DDS device will add a certain amount of jitter. Since every clock will already have an intrinsic level of jitter, choosing an oscillator with low jitter is critical to begin with. Dividing down the frequency of a high-frequency clock is one way to reduce jitter. With frequency division, the same amount of jitter occurs within a longer period, reducing its percentage of system time.

In general, to reduce essential sources of jitter and avoid introducing additional sources, one should use a stable reference clock, avoid using signals and circuits that slew slowly, and use the highest feasible reference frequency to allow increased oversampling.

Spurious-Free Dynamic Range (SFDR) refers to the ratio (measured in decibels) between the highest level of the fundamental signal and the highest level of any spurious, signal including aliases and harmonically related frequency components in the spectrum. For the very best SFDR, it is essential to begin with a high-quality oscillator.

SFDR is an important specification in an application where the frequency spectrum is being shared with other communication channels and applications. If a transmitter's output sends spurious signals into other frequency bands, they can corrupt, or interrupt neighboring signals.

1.4 OBJECTIVE

The main objective of this report is to study all type of architectures of Direct Digital Frequency Synthesizer (DDFS). In this we need to investigate all type of existing structures of DDFS and their speed, area complexity. Because speed and area are two Para meters which are very important to control for an efficient design.

The main part of DDFS is CORDIC (Coordinate Rotation digital computer). CORDIC is an efficient technique to implement DDFS. So we will also study about CORDIC algorithm.

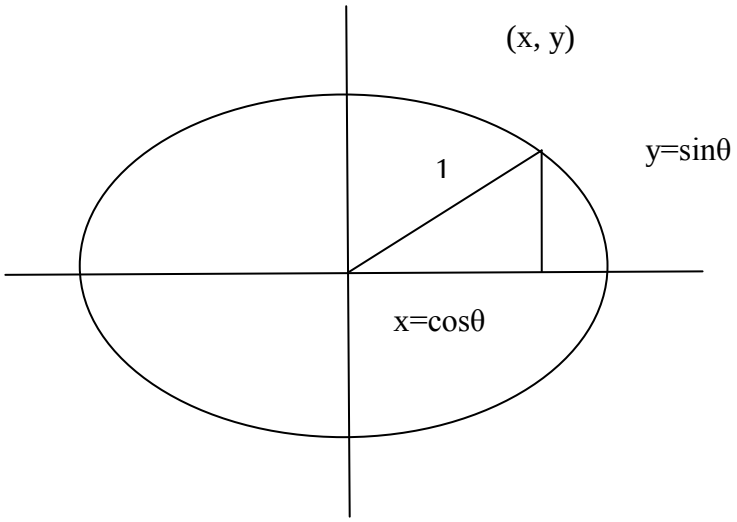
Chapter 2

Coordinate Rotation Digital Computer

2.1 CORDIC ALGORITHM

This algorithm was originally developed by J.E.Volder to compute the rotation of a vector in the Cartesian coordinate system. The method has been extended for computation of hyperbolic functions, multiplications, divisions, exponentials and algorithms [10].

Figure: 2.1 – Unit vector rotation to desired angle



Multiplication by sine and cosine is an integral part of any communication system, so area and time efficient techniques for iterative computation of CORDIC are critical The best application of the cordic algorithm is its use in DDFS(Direct Digital Frequency Synthesizer).To bring a unit vector to desired angle θ, the basic CORDIC algorithm gives known recursive rotations to the vector. Once the vector is at desired angle, then the x and y coordinates of the vector are equal to cosθ and sinθ respectively [11].

Mathematically the angle θ is approximated by addition and subtraction of angles of N successive rotations Δθi. Now approximate expression we have:

$\theta = \sum_{i=0}^{n-1} \sigma i \, \Delta\theta i$ for $\sigma_i = 1$ for positive rotation, $\sigma_i = -1$ for negative rotation

As shown in the figure below, the unit vector in iteration i is rotated by some angle θ_i and the next rotation is made by angle $\Delta\theta_i$ that brings the vector to new angle θ_{i+1} then

$$\text{Cos } \theta_{i+1} = \cos(\theta_i + \sigma_i \Delta\theta_i) = \cos(\theta_i)\cos(\Delta\theta_i) - \sigma_i \sin(\theta i)\sin(\Delta\theta_i) \quad \text{------(1)}$$

$$\text{Sin } \theta_{i+1} = \sin(\theta_i + \sigma_i \Delta\theta_i) = \sin(\theta_i)\cos(\Delta\theta_i) + \sigma_i \cos(\theta i)\sin(\Delta\theta_i) \quad \text{------(2)}$$

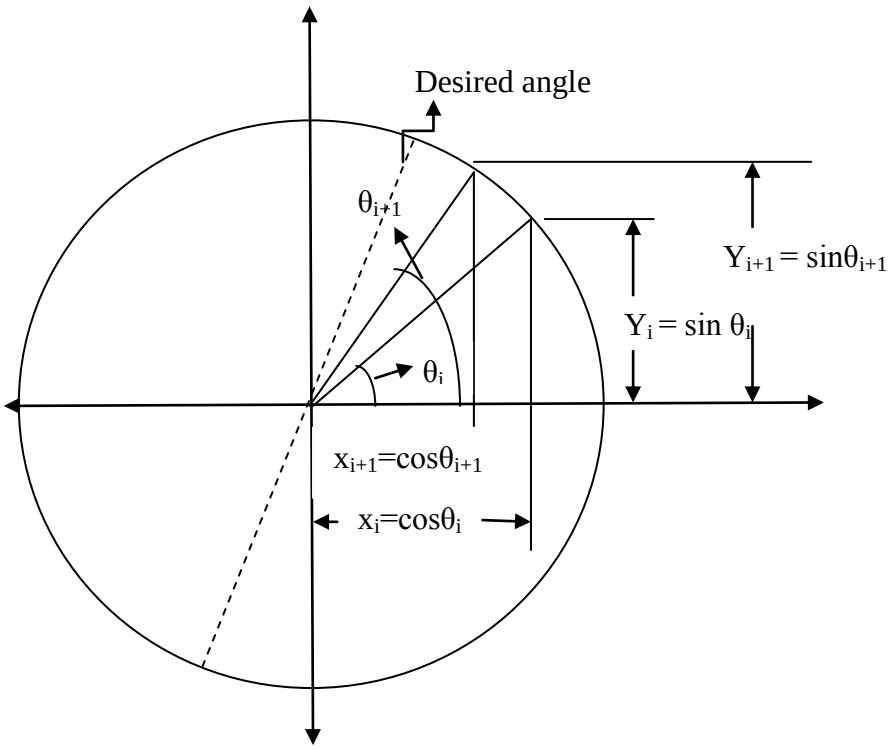
From figure we are having $x_i = \cos(\theta_i)$, $y_i = \sin(\theta_l)$ -----(3)

similarly $x_{i+1} = \cos(\theta_{i+1})$, $y_{i+1} = \sin\theta_{i+1}$ ------(4)

By using equation (3) and (4) in equation (1) and (2)

$$\begin{bmatrix} x\,i+1 \\ yi+1 \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta i) & -\sigma i \sin(\Delta\theta i) \\ \sigma i \sin(\Delta\theta i) & \cos(\Delta\theta i) \end{bmatrix} \begin{bmatrix} xi \\ yi \end{bmatrix} \quad \text{-------(5)}$$

Figure: 2.2 - Incremental rotation by $\Delta\theta i$



Take $\cos(\Delta\theta_i)$ as common

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \cos(\Delta\theta_i) \begin{bmatrix} 1 & -\sigma_i \tan(\Delta\theta_i) \\ \sigma_i \tan(\Delta\theta_i) & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad \text{-------(6)}$$

In this expression $\cos(\Delta\theta_i)$ can also be written in terms of $\tan \Delta\theta_i$ as:

$$\text{Cos } \Delta\theta_i = \frac{1}{\sqrt{1+\tan^2 \Delta\theta i}} \quad \text{---------(7)}$$

To avoid multiplication in computing equation (6), the incremental rotation is set as:

$$\tan \Delta\theta_i = 2^{-i} \quad \text{----------(8)}$$

Which implies $\Delta\theta_i = \tan^{-1} 2^{-i}$, and algorithm applies N such successive micro rotations of $\pm \Delta\theta_i$ to get to the desired angle θ. This put a limit on the desired angle as:

$$\sum_{i=0}^{N-1} \Delta\theta i \leq \theta \leq \sum_{i=0}^{N-1} \Delta\theta i \quad \text{--------(9)}$$

This can be written as a basic rotation of the CORDIC algorithm:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = K_i R_i \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad \text{----------(10)}$$

Where $k_i = \frac{1}{\sqrt{1+2^{-2i}}}$ and $Ri = \begin{bmatrix} 1 & \sigma i * 2-i \\ \sigma i * 2-i & 1 \end{bmatrix}$ --------(11)

Start iteration by putting i=0 in equation (11)

$$\begin{bmatrix} x1 \\ y1 \end{bmatrix} = k_0 R_0 \begin{bmatrix} x0 \\ y0 \end{bmatrix} \quad \text{----------(12)}$$

Put i= 1 in equation (11)

$$\begin{bmatrix} x2 \\ y2 \end{bmatrix} = k_1 R_1 \begin{bmatrix} x1 \\ y1 \end{bmatrix} \quad \text{---------(13)}$$

From (12) and (13) we have :

$$\begin{bmatrix} x2 \\ y2 \end{bmatrix} = k_0 k_1 R_0 R_1 \begin{bmatrix} x0 \\ y0 \end{bmatrix} \qquad \text{---------(14)}$$

Similarly for i=N-1 y0

$$\begin{bmatrix} xN \\ yN \end{bmatrix} = k_0 k_1 \ldots k_{N-1} R_0 R_1 \ldots R_{N-1} \begin{bmatrix} x0 \\ y0 \end{bmatrix} \qquad \text{--------(15)}$$

We can find product of all k as:

$$k = k0k1k2\ldots\ldots kN\text{-}1 = \rightarrow \prod_{i=0}^{N-1} \frac{1}{\sqrt{1+2^{-2i}}} \qquad \text{---------(16)}$$

we have rotation of unit vector and we start our iteration by putting x0=1 and y0=0 so now we can get k inside metrics as:

$$\begin{bmatrix} xN \\ yN \end{bmatrix} = R_0 R_1 R_2 \ldots\ldots\ldots R_{N-1} \begin{bmatrix} k \\ 0 \end{bmatrix}$$

Now we are introducing a new equation to trace desired angle that is:

$\theta_{i+1} = \theta_i - \sigma i \tan^{-1} 2^{-i}$

Put θ_i = desired angle and do iteration by putting i=0 to I =N-1 ,we want $\theta_{i+1} = 0$ at the end of iteration. Now we are having three equations :

$x_{i+1} = x_i - \sigma_i 2^{-i} y_i$

$y_{i+1} = y_i + \sigma_i 2^{-i} x_i \qquad , \quad \sigma_i = \pm 1$

$\theta_{i+1} = \theta_i - \sigma i \tan^{-1} 2^{-i}$

σ_i is the decision operator and is used to determine the direction of rotation

All $\tan^{-1} 2^{-I}$ are pre computed and store in array ,so final iteration we have

$x_N = \cos \theta_d$

$y_N = \sin \theta_d$

In this way at final Nth iteration we will get sin and cos of the desired angle. With the help of CORDIC we are able to calculate multiplications, divisions, hyperbolic , exponentials and logarithms functions.

2.2 PSEUDO ROTATIONS

By **t**aking out the factor cosθ term we can rewrite the equation for I =1 iteration:

$X2 = x1 \cos\theta - y1 \sin\theta = \cos\theta (x1 - y1 \tan\theta)$

$Y2 = x1 \sin\theta + y1 \cos\theta = \cos\theta (y1 + x1 \tan\theta)$

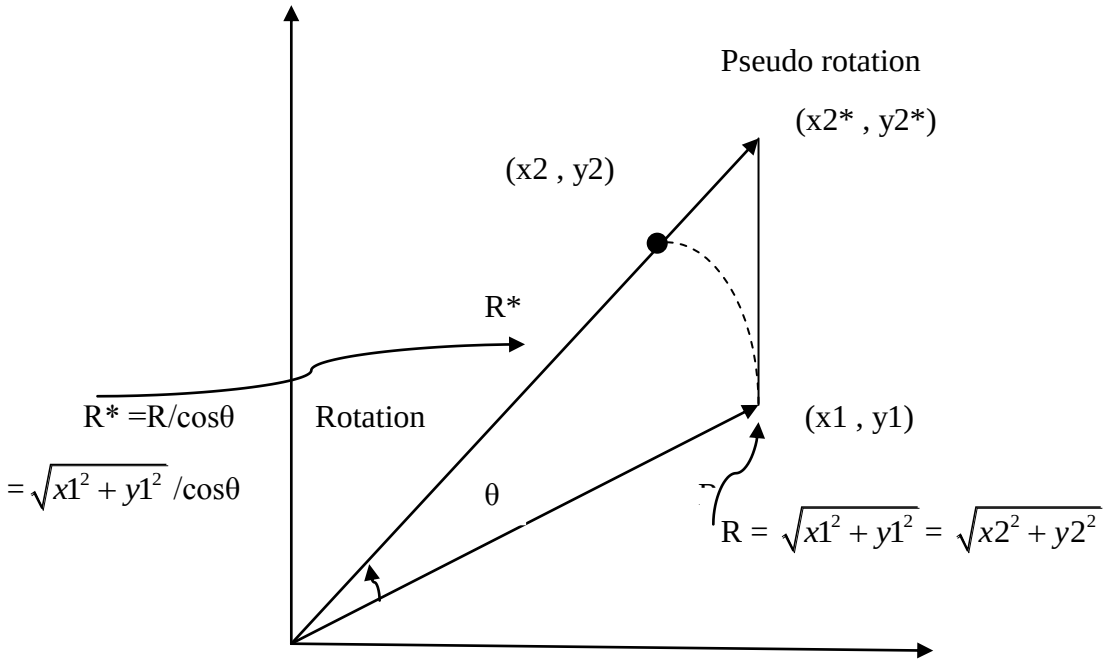
If we drop the cosθ term then we have a pseudo-rotation:

$X2^* = x1 - y1 \tan\theta$

$Y2^* = y1 + x1 \tan\theta$

In this the angle of rotation is correct but the x and y values are scaled by $\cos^{-1}\theta$ (i.e. both become larger than before as $\cos^{-1} \theta > 1$). By dropping cosθ term our computation become easy that is why we drop this term.

Figure: 2.3 – Pseudo rotation



2.3 SCALING FACTOR

The scaling is a product of the pseudo rotations. When we simplify the algorithm to allow pseudo–rotations then cosθ term was left. Thus x and y values are scaled by a scaling factor k where:

$$K = \prod_{n} \frac{1}{\sqrt{1 + 2^{-2i}}}$$

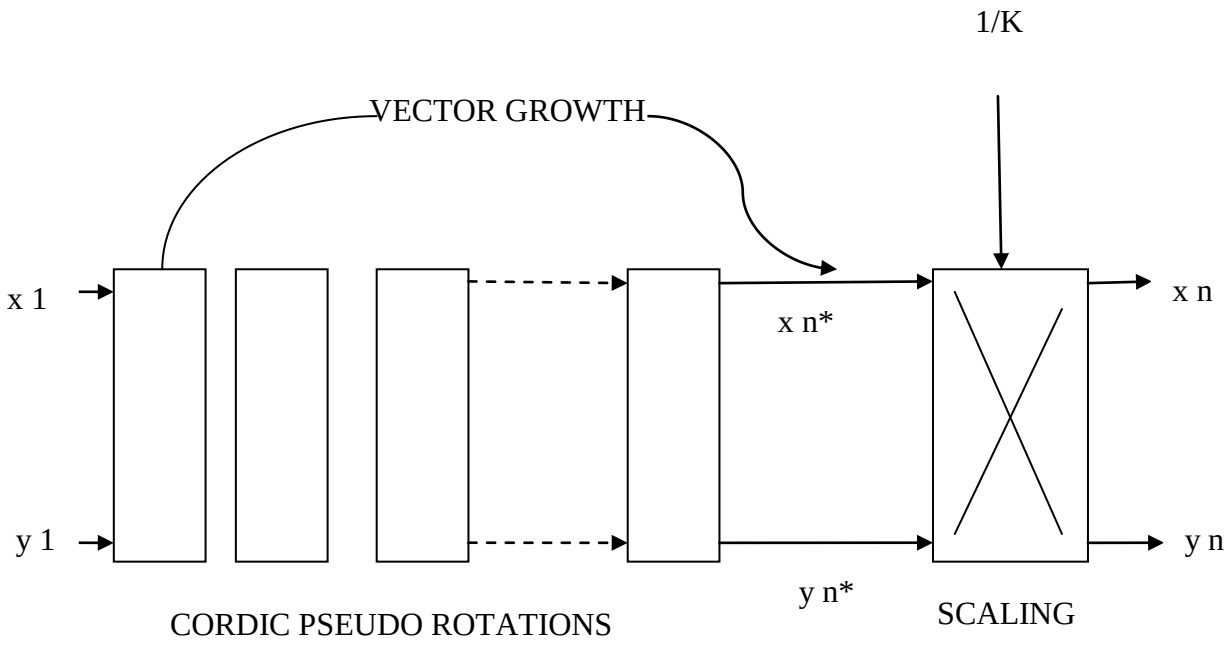
Now if we know the number of iterations then the scaling factor can be precompiled. Also 1/k can be precompiled and used to calculate the true value of x and y.

$K \rightarrow 1.6476$ as $n \rightarrow \infty$

$1/K \rightarrow 0.6073$ as $n \rightarrow \infty$

n = number of iterations

Figure: 2.4 – Solution to pseudo rotation



Now we are showing first 13 CORDIC rotation angles:

Table: 2.1 - CORDIC Rotation Angles

Iteration (i)	Tan(θ)	Angle($\Delta\theta_i$)
0	1	45.000000
1	0.5	26.565051
2	0.25	14.036243
3	0.125	7.1250163
4	0.0625	3.576334
5	0.03125	1.789910
6	0.015625	0.895173
7	0.0078125	0.447614
8	0.00390625	0.223810
9	0.001953125	0.111905
10	0.000976563	0.055952
11	0.000488281	0.027976
12	0.000244141	0.013988

Angles can be rotated in the range $-99.7 \leq \theta_i \leq +99.7$, as the sum of all angles obeying the law $\tan \Delta\theta_i = 2^{-i}$ is 99.7.For angles outside this range trigonometric identities can be used to convert the desired angle into one within the range.

2.4 CORDIC OPERATION MODES

In CORDIC algorithm we are having two modes:

(1) Rotation mode

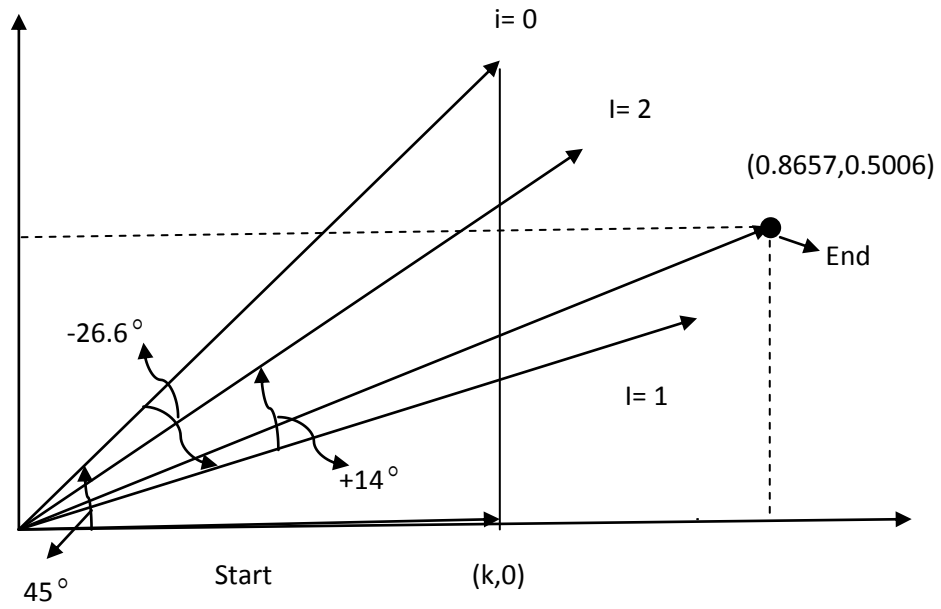
(2) Vectoring mode

2.4.1 Rotation mode : In rotation mode θ_i variable is initialised with the value of desired angle, cordic iteration drives the vector towards this angle, through a series of smaller micro rotations. The direction of rotation is determined by the decision operator σ_i by comparing θ_{i+1} with zero. The θ_{i+1} is converges towards the zero as rotation converges towards desired angle. Now we see how the iteration is going while calculating sin and cosine value of angle $30°$.

Table: 2.2 – Iteration to calculate sine and cosine 30 degree

Iteration(i)	σi	Angle($\Delta\theta_i$)	θ_i	y_i	x_i
0	+1	45	+30	0	0.6073
1	-1	26.6	-15	0.6073	0.6073
2	+1	14	+11.6	0.3036	0.9109
3	-1	7.1	-2.4	0.5313	0.8350
4	+1	3.6	+4.7	0.4270	0.9014
5	+1	1.8	+1.1	0.4833	0.8747
6	-1	0.9	-0.7	0.5106	0.8596
7	+1	0.4	+0.2	0.4972	0.8676
8	-1	0.2	-0.2	0.5046	0.8637
9	+1	0.1	+0	0.5006	0.8657

Figure: 2.5 - Iteration to calculate sine and cosine of angle 30°



Hardware mapping: The CORDIC algorithm exhibits natural affinity for hardware mapping. Each iteration of the algorithm can be implemented as a cordic element(CE). A cordic element is shown in figure given below:

Figure: 2.6 - A CORDIC element

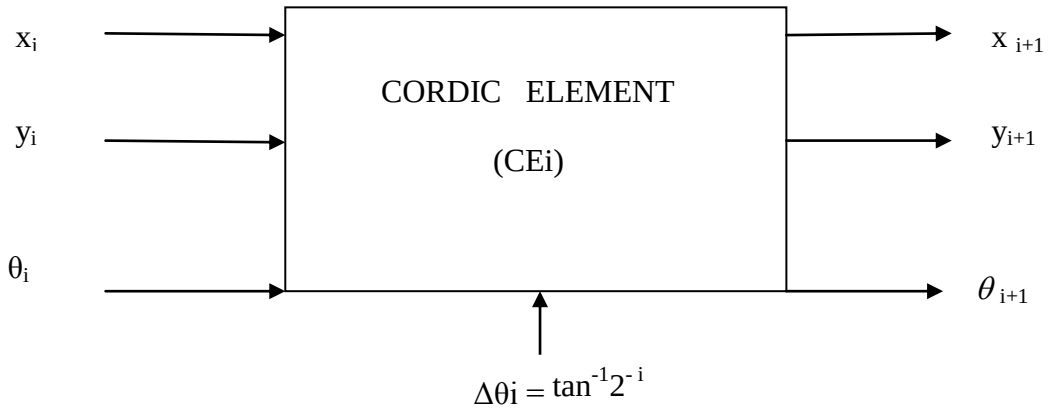
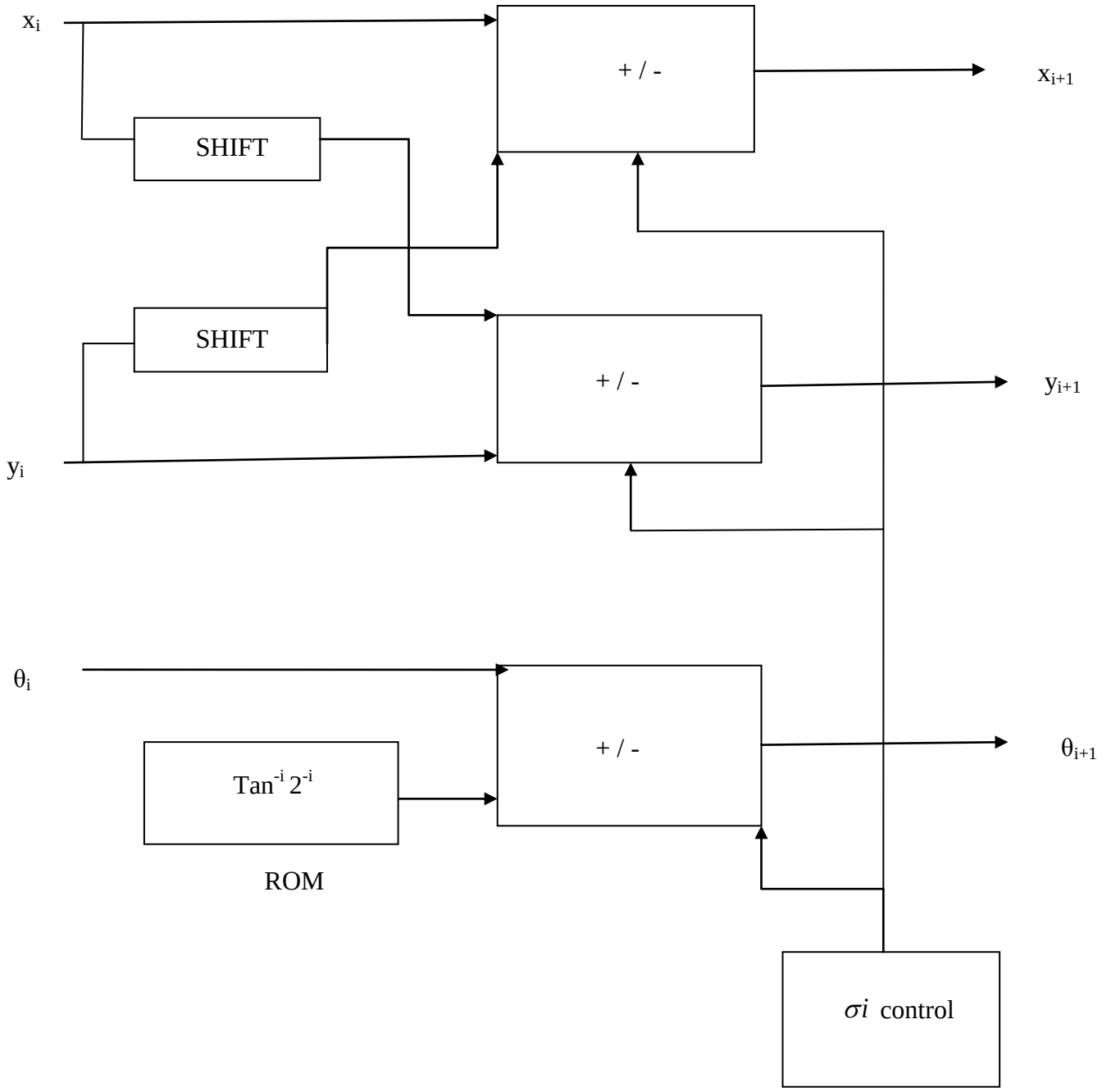


Figure: 2.7 - Implementation of ith iteration of the CORDIC element



2.4.2 Vectoring mode: In this mode input vector is rotated onto x axis. So the direction of rotation is determined through the y input. If y>0 then vector is above x axis and must be rotated downward, if y<0 then vector is below x axis and must be rotated upward. This mode is used to calculate $\tan^{-1}$ y function. So in circular coordinate system we can calculate sin cosine and $\tan^{-1}$. In Vectoring mode $y_{i+1} \rightarrow 0$ (converges to zero) and σi = - sign (x_i y_i)

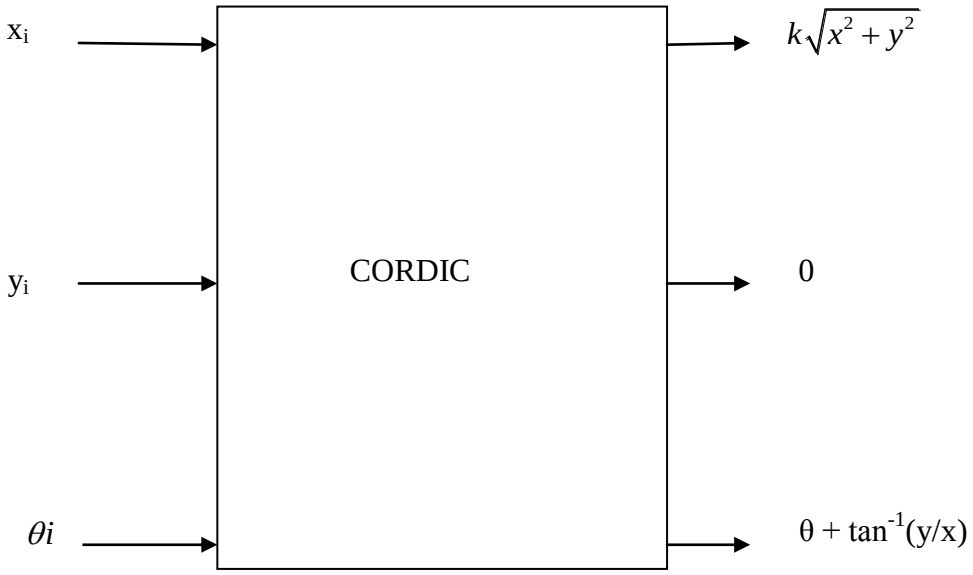


Figure: 2.8 - Cordic in Vectoring Mode

For $\tan^{-1}$ y , put x_0 =1 and $\theta_0 = 0$

In rotation mode

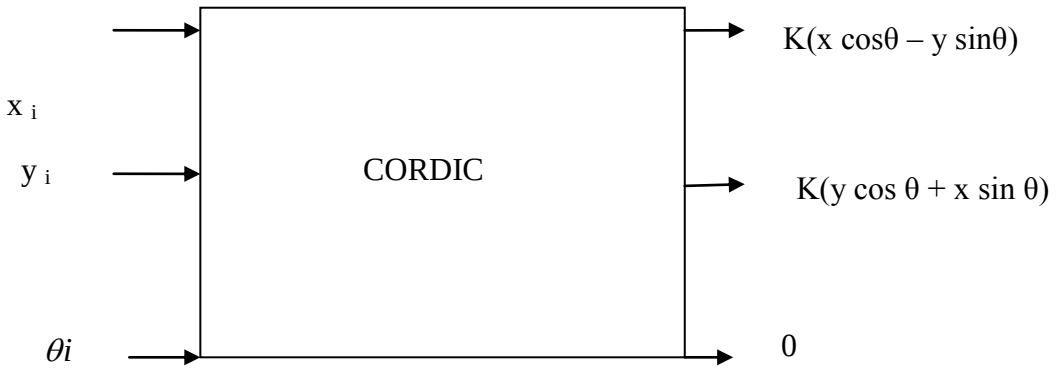


Figure: 2.9 - Cordic in Rotation Mode

Circular coordinate system: We are limited to the number of functions that we can calculate with rotation in circular coordinate. By considering rotations in the other coordinate system we can calculate more functions directly such as multiplications and divisions, which allow us to calculate even more functions indirectly.

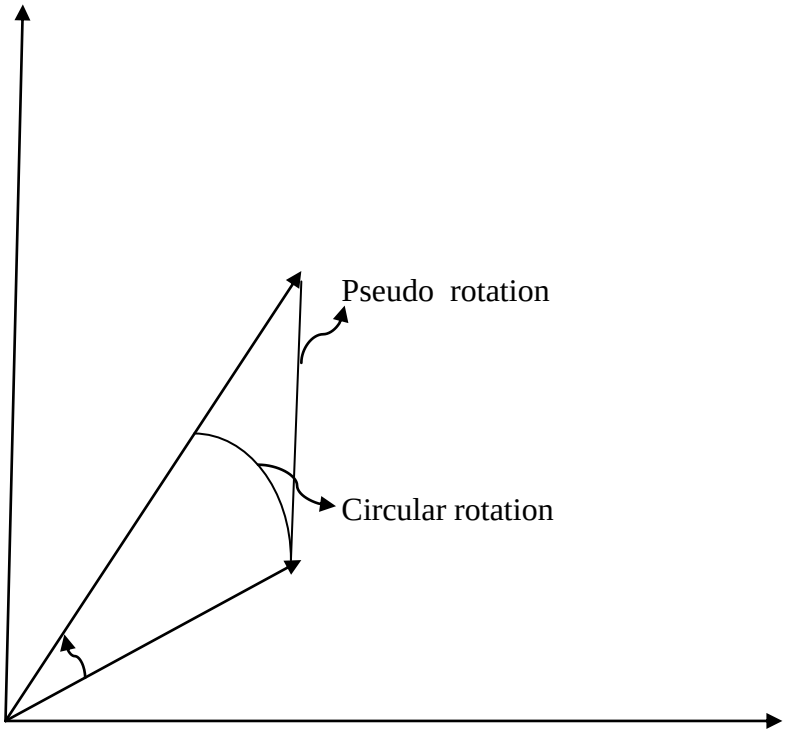


Figure: 2.10 - Rotation in Circular Coordinate System

Other coordinate systems : we are having two more coordinate system. These are:

(1) Linear Coordinate System

(2) Hyperbolic Coordinate System

Figure 2.11 - Linear Coordinate System

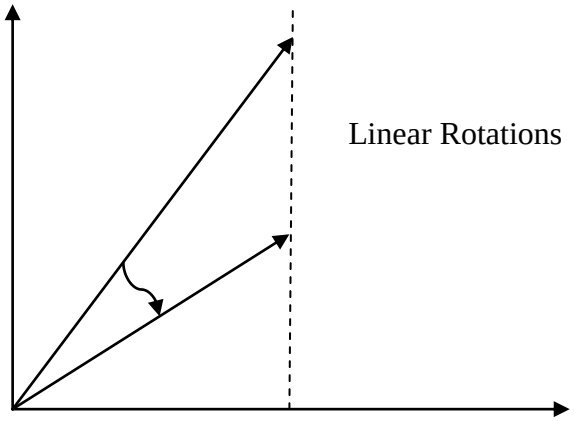
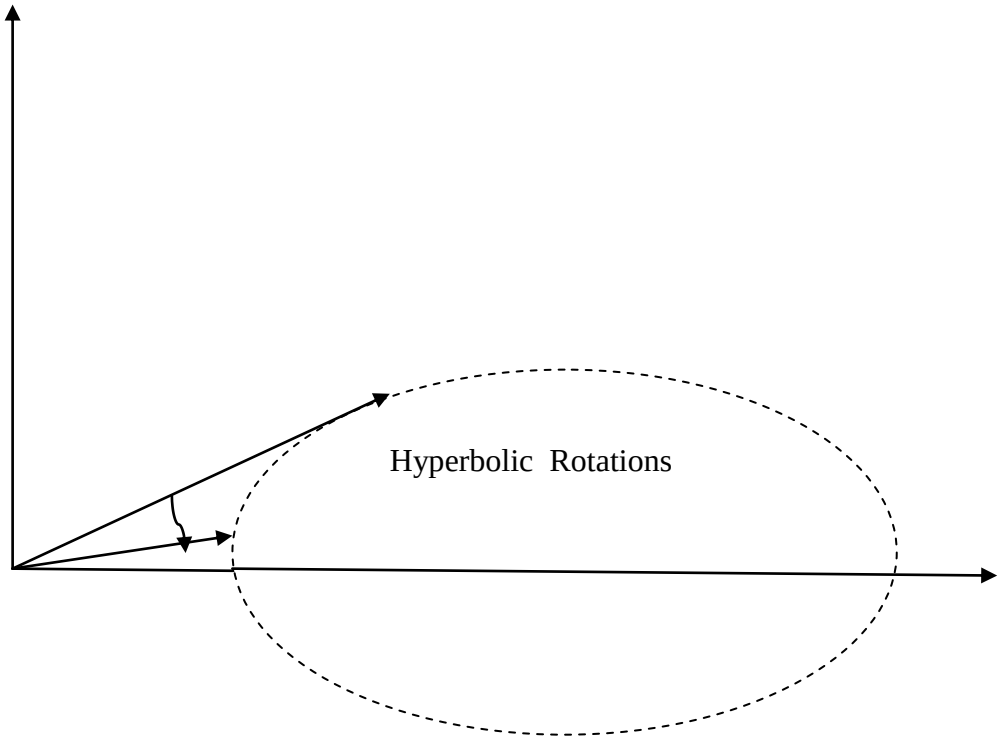


Figure: 2.12 - Hyperbolic Coordinate System



2.5 GENERALIZED CORDIC

With the addition of two other coordinate system , the CORDIC equations can now be generalised to accommodate all three coordinate systems:

$x_{i+1} = x_i - \mu(i)\, d_i\, (2^{-i}\, y_i)$

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

$z_{i+1} = z_i - d_i\, (e^{\,i})$

For Circular Rotations: $\mu = 1$, $e^{\,i} = \tan^{-1} 2^{-i}$

For Linear Rotations: $\mu = 0$, $e^{\,i} = 2^{-i}$

For Hyperbolic Rotations: $\mu = -1$, $e^{\,i} = \tanh^{-1} 2^{-i}$

2.6 DIFFERENCE BETWEEN CIRCULAR AND OTHER COORDINATE SYSTEMS

The advantage of using other coordinate systems with the CORDIC algorithm is that it allows more functions to be calculated. The drawback is that the system becomes more complex. The set of rotation angles used for the circular coordinate system are no longer valid when using the CORDIC algorithm with a linear or hyperbolic system. Hence, two other sets of angles are used for rotation s made in these systems. It is noticeable that, when using the CORDIC algorithm for hyperbolic rotations, the scaling factor k is different from that used for circular rotations.

The hyperbolic scaling factor is denoted by k* and is calculated using the equation:

$$K^*_n = \prod_n \left(\sqrt{1 + 2^{-2i}}\right)$$

$K^*_n \rightarrow 0.82816$ as $n \rightarrow \infty$

$1/\, k^*_n \rightarrow 1.20750$ as $n \rightarrow \infty$

n= number of iterations

The linear and hyperbolic co-ordinate systems are different in a number of respects

- Shift sequence
- Angles of convergence
- Scaling factor

Co-ordinate System	μ	Scale Factor ($n \to \infty$)	Convergence(max. angle as $n \to \infty$)	Shift Sequence
Circular	+1	1.64676	99.7°	0,1,2,3,4,…n-1
Linear	0	1.0	57.3°	1,2,3,4,5,….n
Hyperbolic	-1	0.83816	64.7°	1, 2, 3, 4, 4,5, ….

Table: 2.3 - Scale Factor, Converging angle, shifting sequence of all three coordinate system

From above table it is clear that scale factor, converging angle and shifting sequence of all three co-ordinate systems like circular, linear and hyperbolic are different. Due to this all these three systems are used to calculate different functions. Now with the help of all three co-ordinate system we are able to calculate many more functions like multiplications, divisions, logarithms, exponential etc.

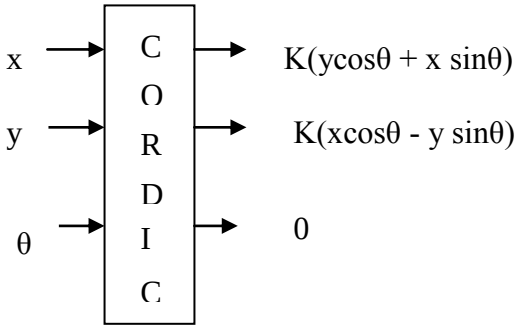
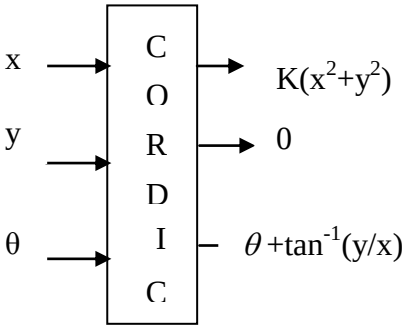
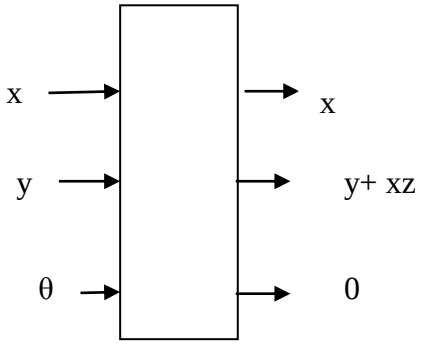
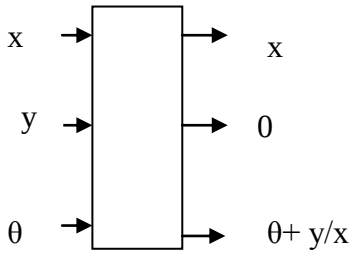
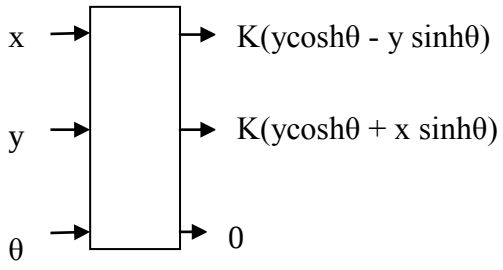
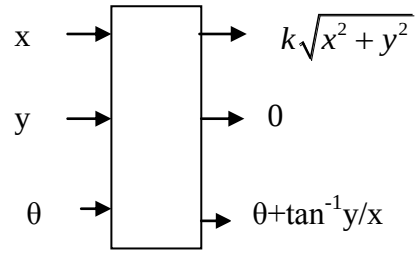
	Rotation mode: σi =sign(θ(i)) θ(i) $\rightarrow$ 0	Vectoring mode: σi = -sign(x(i)y(i)) y(i) $\rightarrow$ 0
μ =1 Circular e(i)= $\tan^{-1} 2^{-i}$	 For cos & sin, set x=1/k, y=0	 For tan-1,set x=1,θ=0
μ =0 Linear e(i)= 2^{-i}	 For multiplications put y=0	 For division put θ=0
μ =-1 Hyperbolic e(i)= $\tanh^{-1}$ 2^{-i}	 For cosh & sinh, set x=1/k* , y=0	 For $\tanh^{-1}$, set x=1, z=0

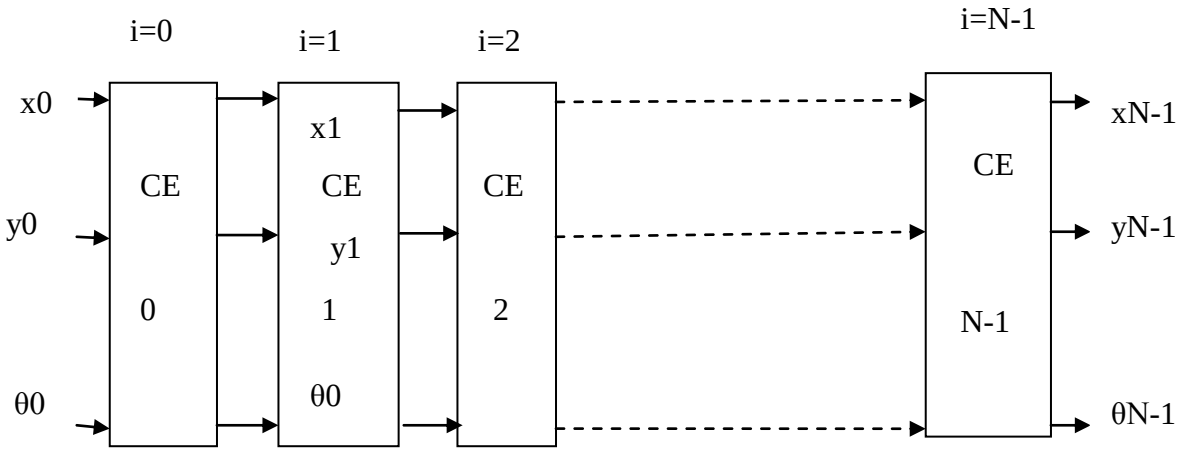
Table: 2.4 - CORDIC modes

2.7 DIFFERENT ARCHITECTURE FOR CORDIC ALGORITHM: We are having many type of architecture to implement CORDIC algorithm on FPGA [11]. Our selection of architecture depends upon various type of requirements like speed and area. Here we will discuss about three types of architecture and these are:

(1) Cascaded architecture
(2) Pipelined architecture
(3) Folded architecture

2.7.1 Cascaded architecture: It consists of array of cells and each cell is used for single iteration.

Figure: 2.13 – Cascaded architecture of CORDIC



In above diagram each block is represented by CE (Cordic Element).This design can be pipelined for better performance .In above architecture we are having more delay. If we will use buffer in between each cell then speed will be increased.

2.7.2 **Pipelined architecture:** If will use buffer in between each block in cascaded architecture then this structure will be called pipelined architecture. The main difference

between cascaded and pipelined architecture is of speed. The speed of pipelined architecture is more. It will increase latency of system.

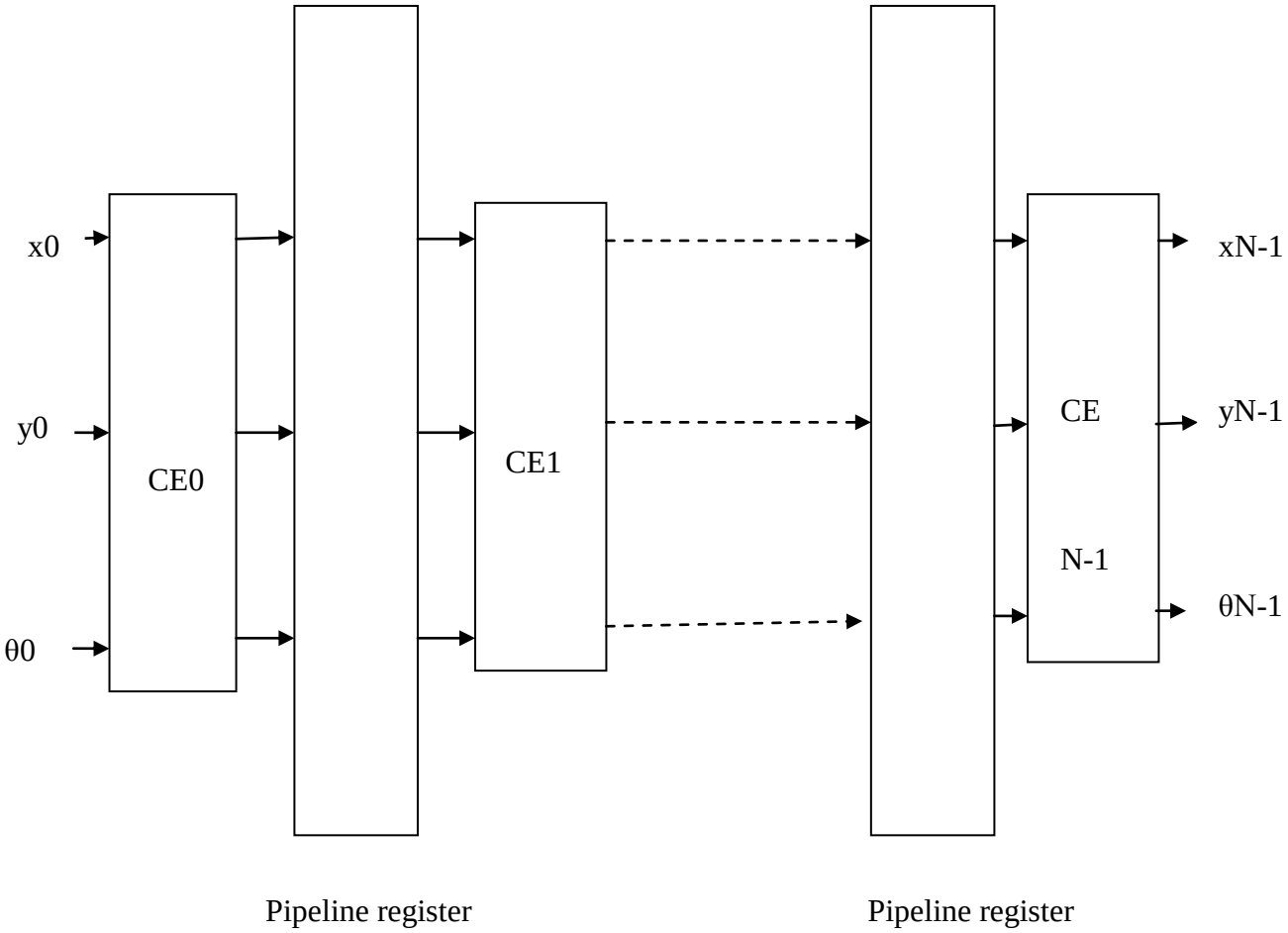
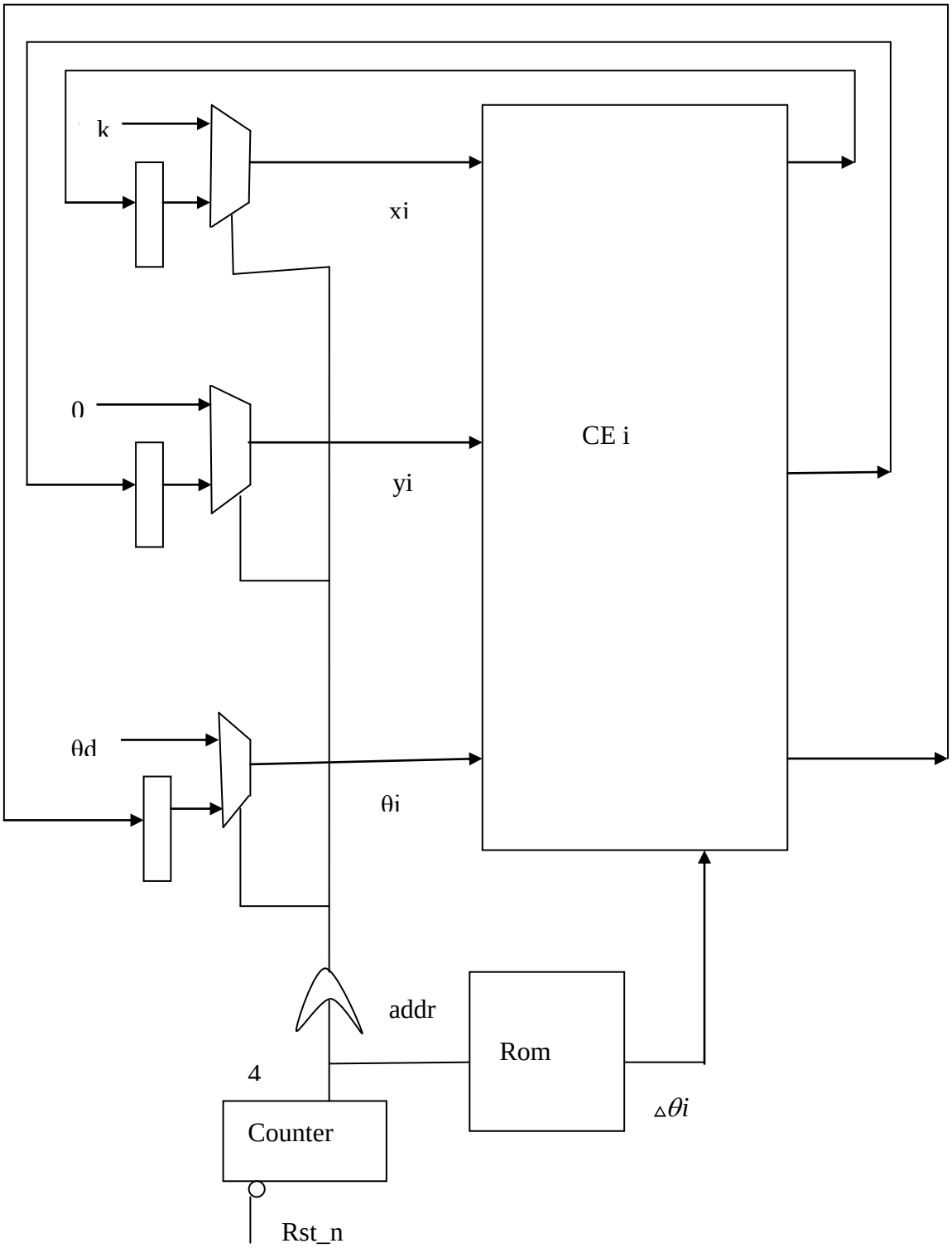


Figure: 2.14 - Pipelined architecture

2.7.3 Folded architecture: If we want to implement our design in less area then we use folded architecture. Basically folding is a time multiplexed architecture. The cordic algorithm is very regular and a folded architecture can be easily crafted. Depending on the number of cycles available for computing the sine and cosine values, the CORDIC algorithm can be folded by any factor. For N=16, the architecture is folded by a folding factor of 16.

Figure: 2.15 - Folded CORDIC Architecture for Folding factor N=16



2.8 APPLICATIONS OF CORDIC

CORDIC is an acronym for CO-ordinate Rotation Digital Computer and was developed by J.E.Volder in 1959.The CORDIC algorithm is an example of an approach that is quite different from traditional math and which is very efficient. The concept of CORDIC is to take an angle θ and rotate a vector over this angle towards zero in a series of steps such that the sum of all the steps taken equal to θ and we can accumulate corresponding X and Y increments for each step such that when we complete the process, Y/X=tan(θ).Variations of the basic concept can result in easy calculation of all the forward and inverse trigonometric functions as well as square root, hyperbolic trigonometric functions, and exponential and logarithmic functions

SIN & COS Function

CORDIC can be used to compute sin of any angle θ with little variation. The angle is given as input. A vector of length 1.647 along the x-axis is taken. The vector is then rotated in steps so as to reach the desired input angle θ. The x and y values are accumulated. After fixed number of iterations the final coordinates of the vector i.e. the x and y values give the values of cos and sin respectively of the given angle θ.

TANGENT

Tangent of an input angle can be calculated using CORDIC by dividing the accumulated values of Y and X after rotation by desired angle θ.

$Y/X = \tan\theta$

INVERSE TRIGONOMETRIC FUNCTIONS

Give the values of x and y, the inverse trigonometric functions can also be calculated. To get the inverse tangent following initial values of constants can be taken.

Set k =0

$\theta = 0$

X =1

Value of Y is given as input. The vector is rotated using CORDIC iterations in steps until the final value of Y is reached equal to zero. The angle during this rotation is the inverse tangent of the value given to Y.

CHAPTER 3

DIRECT DIGITAL FREQUENCY SYNTHESIZER

3.1 INTRODUCTION

The Direct Digital Frequency Synthesizer module is the heart of DDFS function generators. DDFS module produces a digital staircase approximation of a sinusoid in order to construct the sine wave. Generated samples are converted to analog signal and filtered to get pure wave at desired frequency. The staircase approximation of sinusoid is illustrated in figure: 3.1.

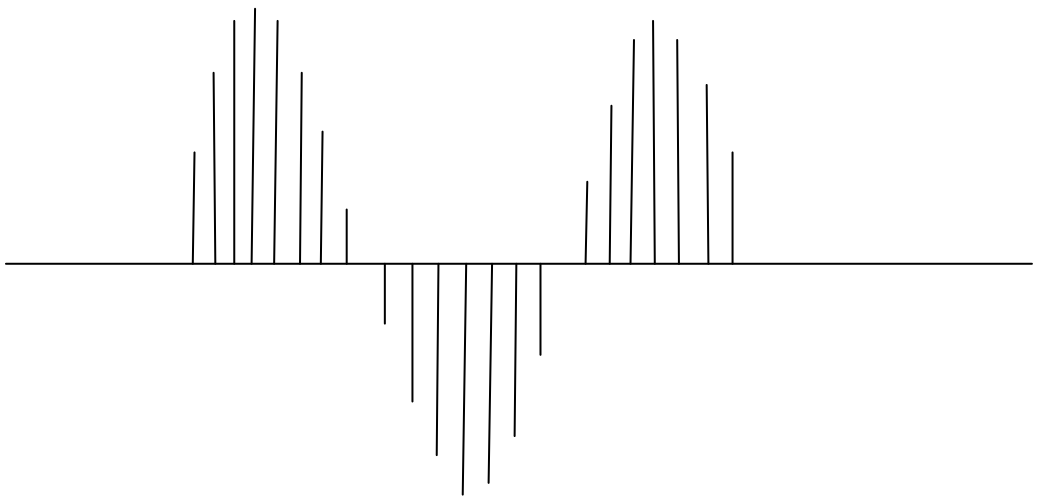


Figure: 3.1 –Staircase of sinusoid

The DDFS modules basically constitute of three main blocks. These are Numerically Controlled Oscillator (NCO), Sine Look-up Table and Digital to Analog Converter (DAC). The NCO comprises the increment register and phase accumulator logic. The increment register stores the binary value of frequency control register. The phase accumulator adds the phase increment value to its accumulator output value. The calculated accumulator output is used to address the look-up table which outputs the digital sample values of sine wave at current phase value.

3.2 BASIC DDFS DESIGN

The main discussion is done on the generation of digital samples according to the desired frequency. The basic DDFS data flow diagram is illustrated in Figure: 3.2. The presented figure include digital to analog conversion structure.

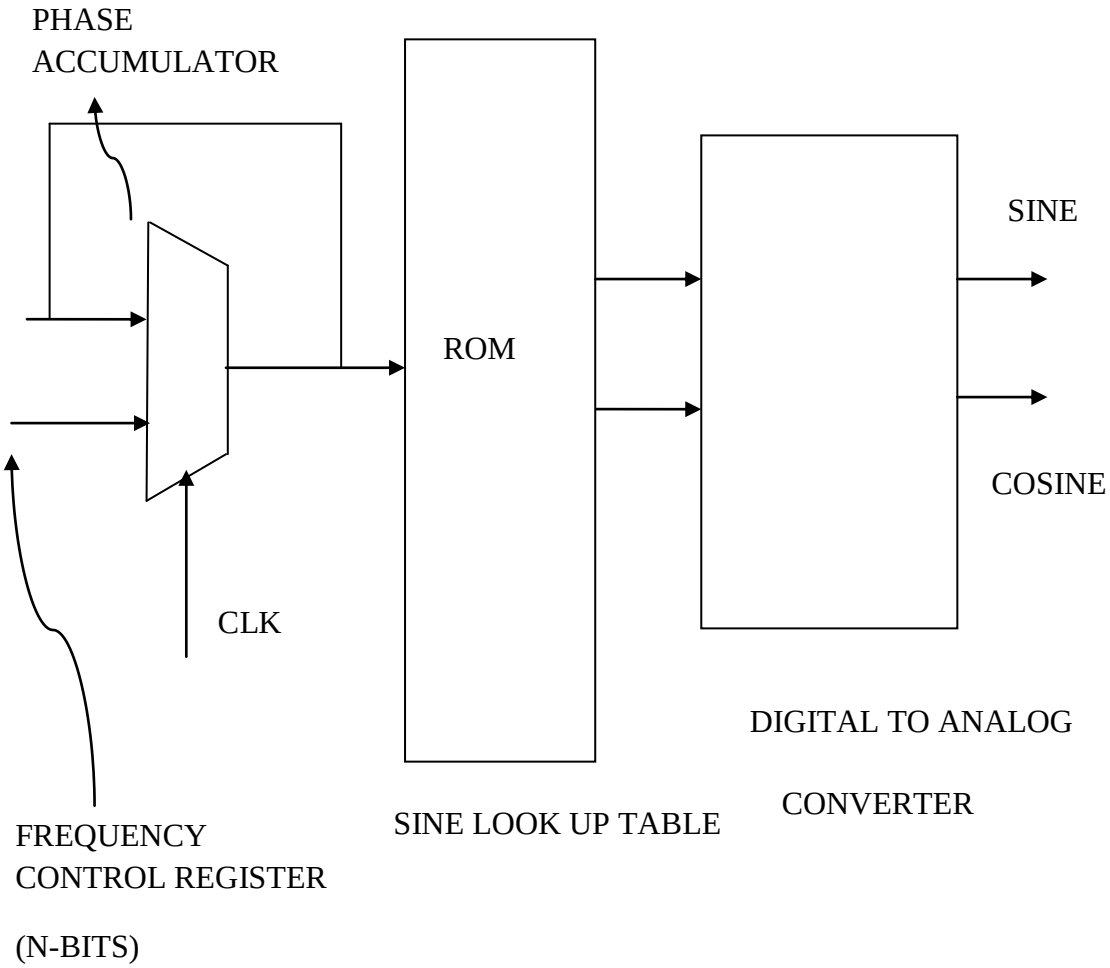


FIGURE:3.2 – Basic DDFS block diagram

At each reference clock cycle, the phase accumulator integrates the phase increment value (frequency control register value) to the phase accumulator output value. The phase increment and accumulator output value are defined by the same number of bits. This frequency control word and used reference clock frequency determine the frequency resolution of DDFS. The complete cycle of sine and cosine are saved in the ROM. The

accumulator generates the address of sine and cosine to access cycle of sine and cosine. The phase accumulator continuously increment the value of address to access next value of sine and cosine.

The frequency of the waveform depends on the reference clock frequency and length of phase accumulator. The waveform frequency is calculated using the formula given below

$Fout = (FCR * Fref) / 2^N$

Where

Fout = output waveform frequency

FCR = phase increment (frequency control register value)

Fref = reference clock frequency

N = phase accumulator word length

3.3 DDFS WITH REDUCED ROM SIZE

The basic design is improved by using the symmetry of sine and cosine waves. The modified design is shown in below Figure. The output of the accumulator is truncated from N to L bits to reduce memory requirement. A complete period 0 to 2π of sine and cosine waves can be generated from values of the two signals from 0 to $\pi/4$. The sizes of the two memories are reduced by one eighth by only storing the values of sine and cosine from 0 to $\pi/4$. The L – 3 bits are used to address the memories, and then three most significant bits (MSBs) of the address are used to map the values to generate complete periods of cosine and sine. A ROM/RAM based DDFS design requires two 2^{L-3} deep memories of width M. The design takes up a large area and dissipates significant power. So we do not want to use large ROM size because only then we can reduce power and area. Several algorithms and techniques have been proposed that reduce or completely eliminate look up tables in memories. A major advantage of a direct digital frequency synthesizer (DDFS) is that its output frequency, phase and amplitude can be precisely and rapidly manipulated under digital processor control. Other inherent DDFS attributes include the ability to tune with extremely fine frequency and phase resolution, and to rapidly "hop" between frequencies. These combined characteristics have made the technology popular in military radar and communications systems.

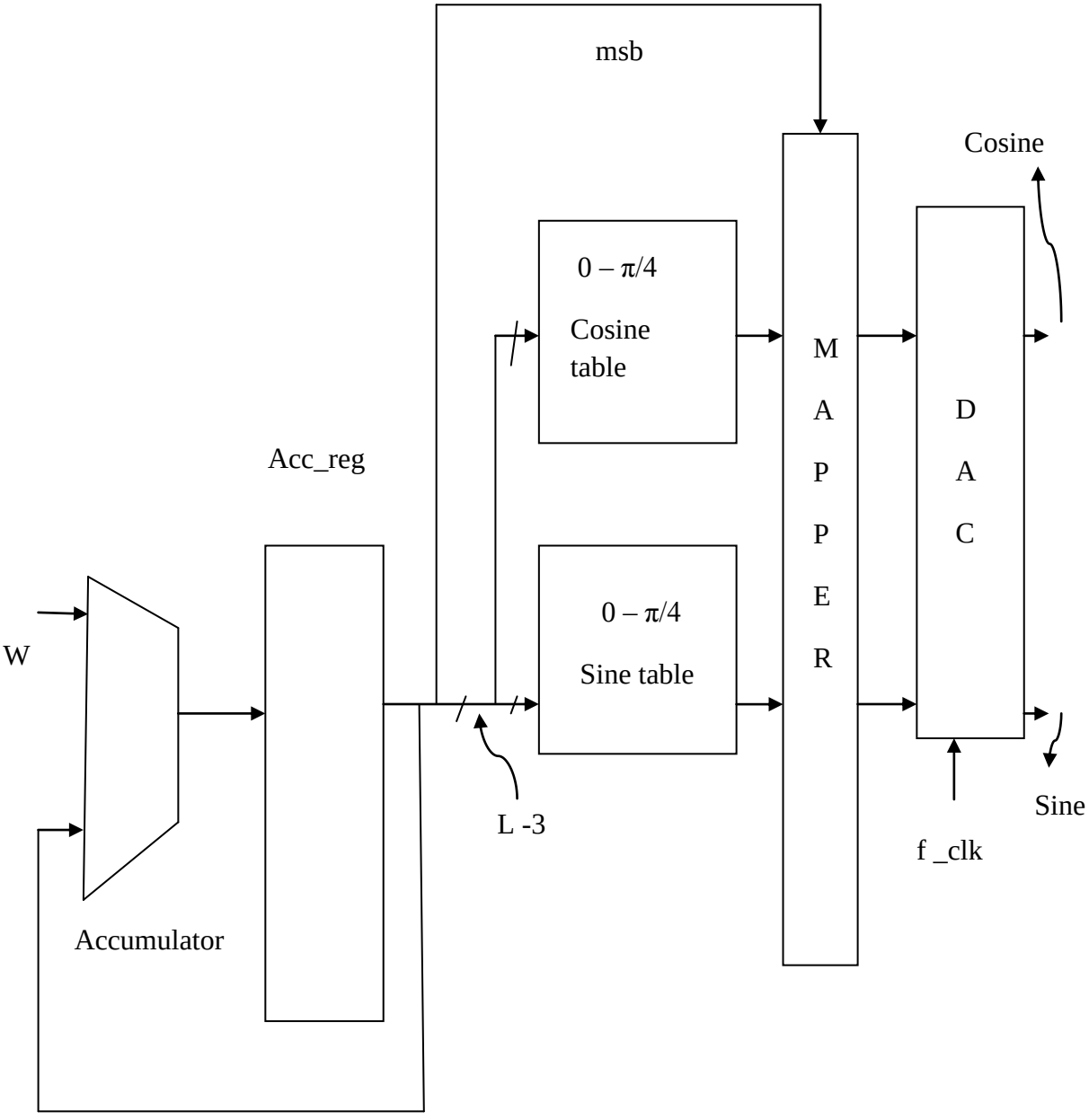
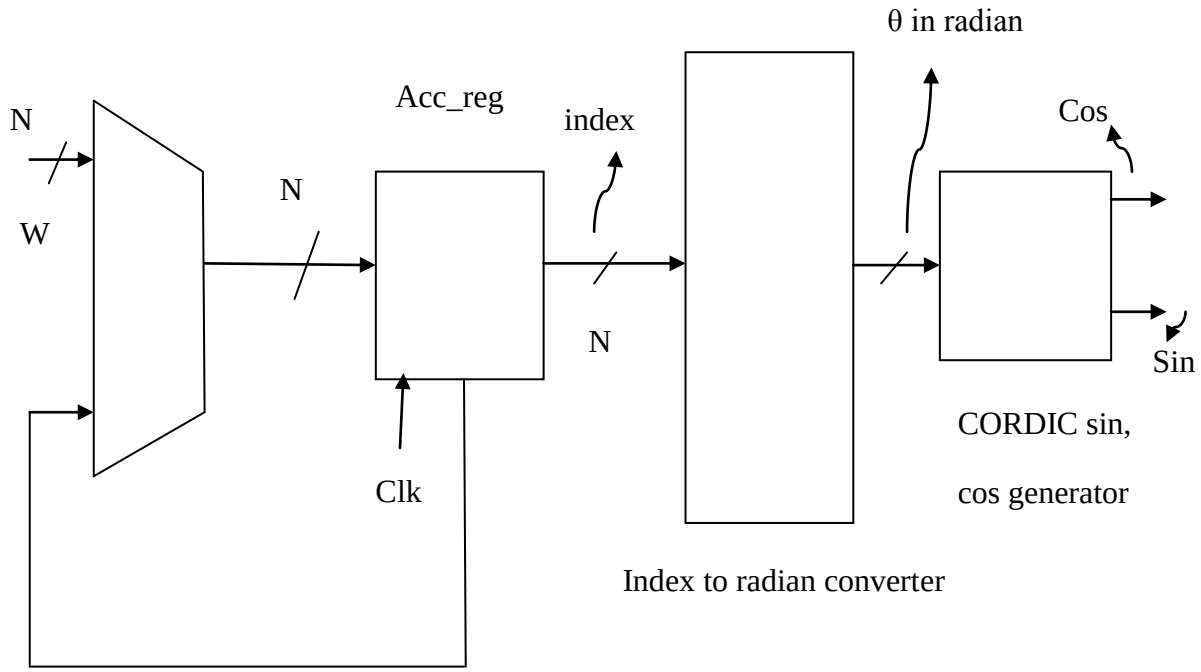


Figure: 3.3 - DDFS WITH REDUCED ROM SIZE

3.4 CORDIC BASED DDFS

The circular-mode CORDIC (coordinate rotation digital computer) algorithm is an iterative method to compute sine/cosine values, in which an initial vector is rotated by a predetermined sequence of sub-angles in such a way that the summation of the rotations approaches the given angle. CORDIC has been a widely-adopted method for implementing the sine/cosine generator in DDFS. When compared to the ROM look-up-table approach, where all the required sine/cosine sample values are stored in a ROM, the CORDIC approach does not lead to the exponential growth of the hardware. The Figure of cordic based DDFS is shown in figure: 3.4

Figure: 3.4 - DDFS with CORDIC block



The CORDIC algorithm takes angle θ in radians, whereas the DDFS accumulator specifies the angle as an index value. To use a CORDIC block , a multiplier is required that converts index n to angle θ in radians, where:

$\theta = (2\pi/2^N) * \text{index}$

3.4.1 DDFS WITH PIPELINED CORDIC BLOCK

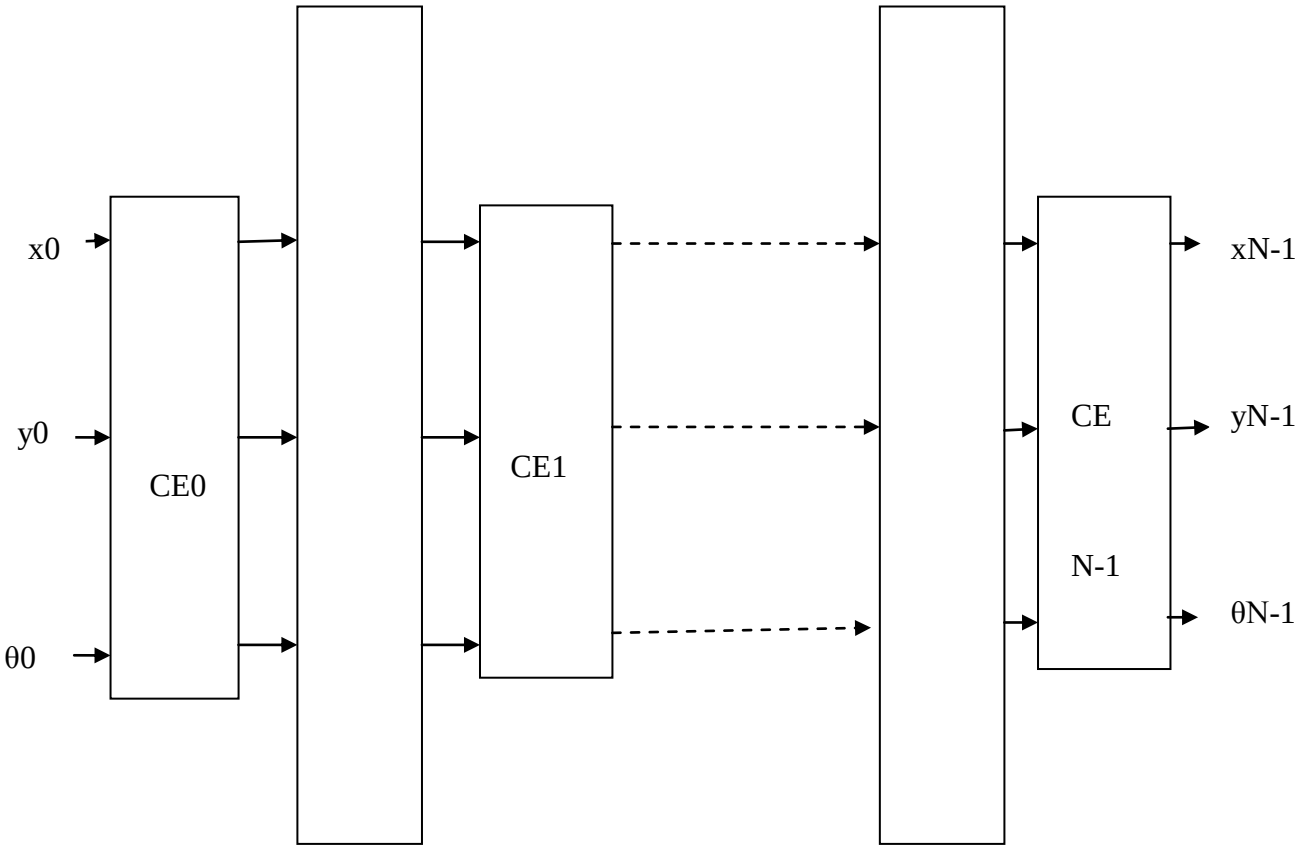
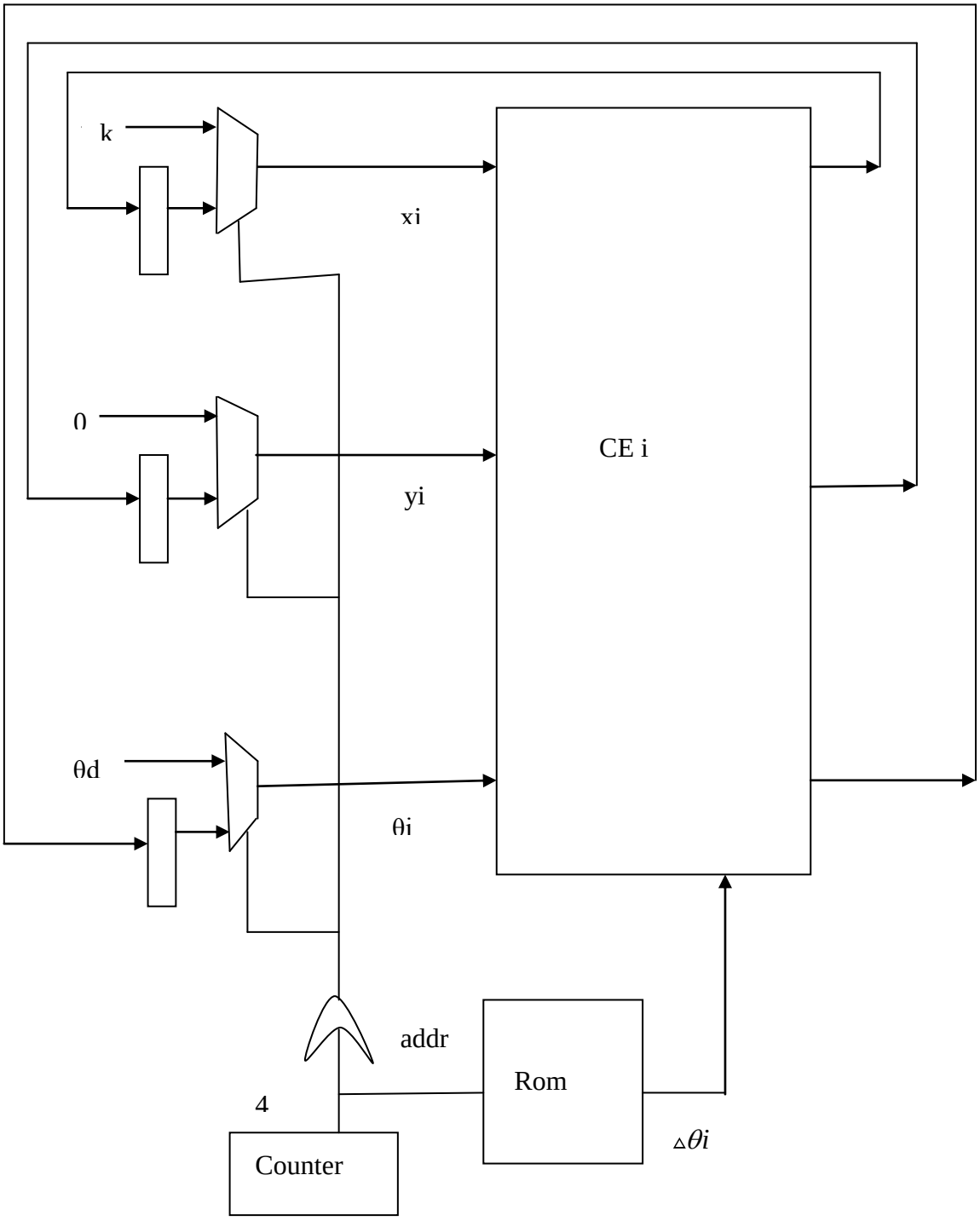


Figure: 3.5 - Pipelined architecture

In this we replace our standard CORDIC with pipelined CORDIC, this will increase the speed of DDFS. But the main drawback of pipelined structure is increased latency. It will just increase the hardware that lead to increase in area.

3.4.2 DDFS WITH FOLDED CORDIC BLOCK

Figure: 3.6 - Folded CORDIC Architecture for Folding factor N=16



Folded CORDIC will decrease the speed, but it is reducing the area. So there is trade off between speed and area.

CHAPTER 4

SIMULATION RESULTS AND DISCUSSIONS

4.1 CORDIC ALGORITHM IMPLEMENTATION

Cordic algorithm was originally developed by J.E.Volder to compute the rotation of a vector in the Cartesian coordinate system. The method has been extended for computation of hyperbolic functions, multiplications, divisions, exponentials and algorithms. `

Multiplication by sine and cosine is an integral part of any communication system, so area and time efficient techniques for iterative computation of CORDIC are critical The best application of the cordic algorithm is its use in DDFS(Direct Digital Frequency Synthesizer).To bring a unit vector to desired angle θ, the basic CORDIC algorithm gives known recursive rotations to the vector. Once the vector is at desired angle, then the x and y coordinates of the vector are equal to cosθ and sinθ respectively.

Now we are introducing a new equation to trace desired angle from chapter 2 that is:

$\theta_{i+1} = \theta_i - \sigma i \tan^{-1} 2^{-i}$

Put θ_i = desired angle and do iteration by putting i=0 to I =N-1 ,we want $\theta_{i+1} = 0$ at the end of iteration. Now we are having three equations :

$x_{i+1} = x_i - \sigma_i 2^{-i} y_i$

$y_{i+1} = y_i + \sigma_i 2^{-i} x_i \quad , \quad \sigma_i = \pm 1$

$\theta_{i+1} = \theta_i - \sigma i \tan^{-1} 2^{-i}$

σ_i is the decision operator and is used to determine the direction of rotation

All $\tan^{-1} 2^{-I}$ are pre computed and store in array ,so final iteration we have

$x_N = \cos \theta_d$

$y_N = \sin \theta_d$

In this way at final Nth iteration we will get sin and cos of the desired angle. With the help of CORDIC we are able to calculate multiplications, divisions, hyperbolic , exponentials and logarithms functions

Angles can be rotated in the range $-99.7 \leq \theta_i \leq +99.7$, as the sum of all angles obeying the law $\tan \Delta\theta_i = 2^{-i}$ is 99.7.For angles outside this range trigonometric identities can be used to convert the desired angle into one within the range.

4.2 CORDIC OPERATION MODES

In CORDIC algorithm we are having two modes:

(1) Rotation mode

(2) Vectoring mode

4.2.1 Rotation mode : In rotation mode θ_i variable is initialised with the value of desired angle, cordic iteration drives the vector towards this angle, through a series of smaller micro rotations. The direction of rotation is determined by the decision operator σ_i by comparing θ_{i+1} with zero. The θ_{i+1} is converges towards the zero as rotation converges towards desired angle. Now we see how the iteration is going while calculating sin and cosine value of angle $30°$.

4.2.2 Vectoring mode: In this mode input vector is rotated onto x axis. So the direction of rotation is determined through the y input. If y>0 then vector is above x axis and must be rotated downward, if y<0 then vector is below x axis and must be rotated upward. This mode is used to calculate $\tan^{-1}$ y function. So in circular coordinate system we can calculate sin cosine and $\tan^{-1}$. In Vectoring mode $y_{i+1} \rightarrow 0$ (converges to zero) and σi = - sign (x_i y_i)

Table: 4.1 – Iteration to calculate sine and cosine 30 degree

Iteration(i)	σi	Angle($\Delta\theta_i$)	θ_i	y_i	x_i
0	+1	45	+30	0	0.6073
1	-1	26.6	-15	0.6073	0.6073
2	+1	14	+11.6	0.3036	0.9109
3	-1	7.1	-2.4	0.5313	0.8350
4	+1	3.6	+4.7	0.4270	0.9014
5	+1	1.8	+1.1	0.4833	0.8747
6	-1	0.9	-0.7	0.5106	0.8596
7	+1	0.4	+0.2	0.4972	0.8676
8	-1	0.2	-0.2	0.5046	0.8637
9	+1	0.1	+0	0.5006	0.8657

Now we are showing first 13 CORDIC rotation angles:

Iteration (i)	Tan(θ)	Angle($\Delta\theta_i$)
0	1	45.000000
1	0.5	26.565051
2	0.25	14.036243
3	0.125	7.1250163
4	0.0625	3.576334
5	0.03125	1.789910
6	0.015625	0.895173
7	0.0078125	0.447614
8	0.00390625	0.223810
9	0.001953125	0.111905
10	0.000976563	0.055952
11	0.000488281	0.027976
12	0.000244141	0.013988

Table: 4.2 - CORDIC Rotation Angles

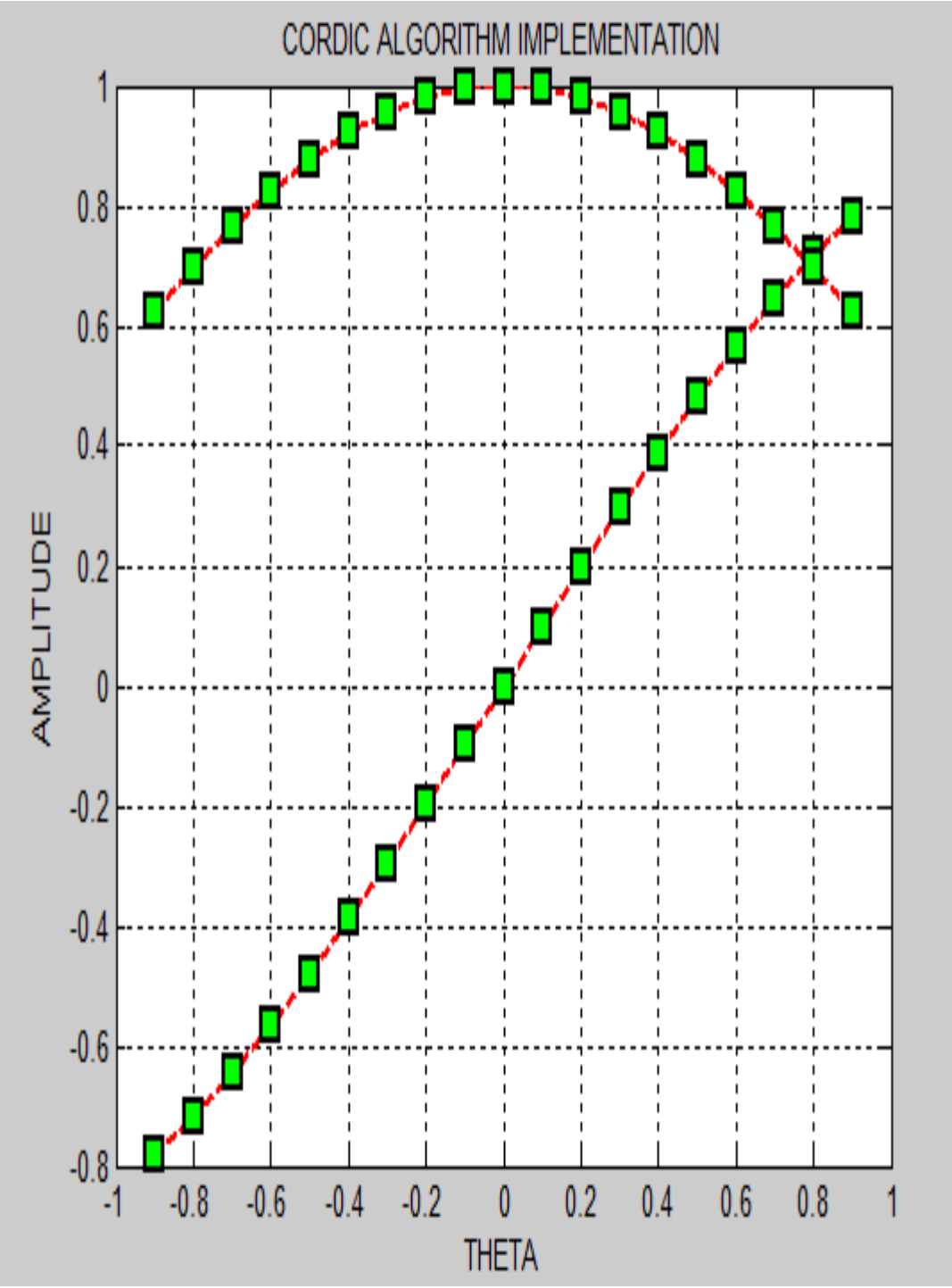


Figure 4.1: - Cordic algorithm implementation of Sine and Cosine from -0.9 radian to +0.9 radian

4.3 CORDIC ALGORITHM IMPLEMENTATION OF SINE AND COSINE FROM 0 TO Pi/4

Initially we are having sine and cosine from -0.9 radian to +0.9 radian, From that we have cut down sine and cosine wave from 0 to pi/4.

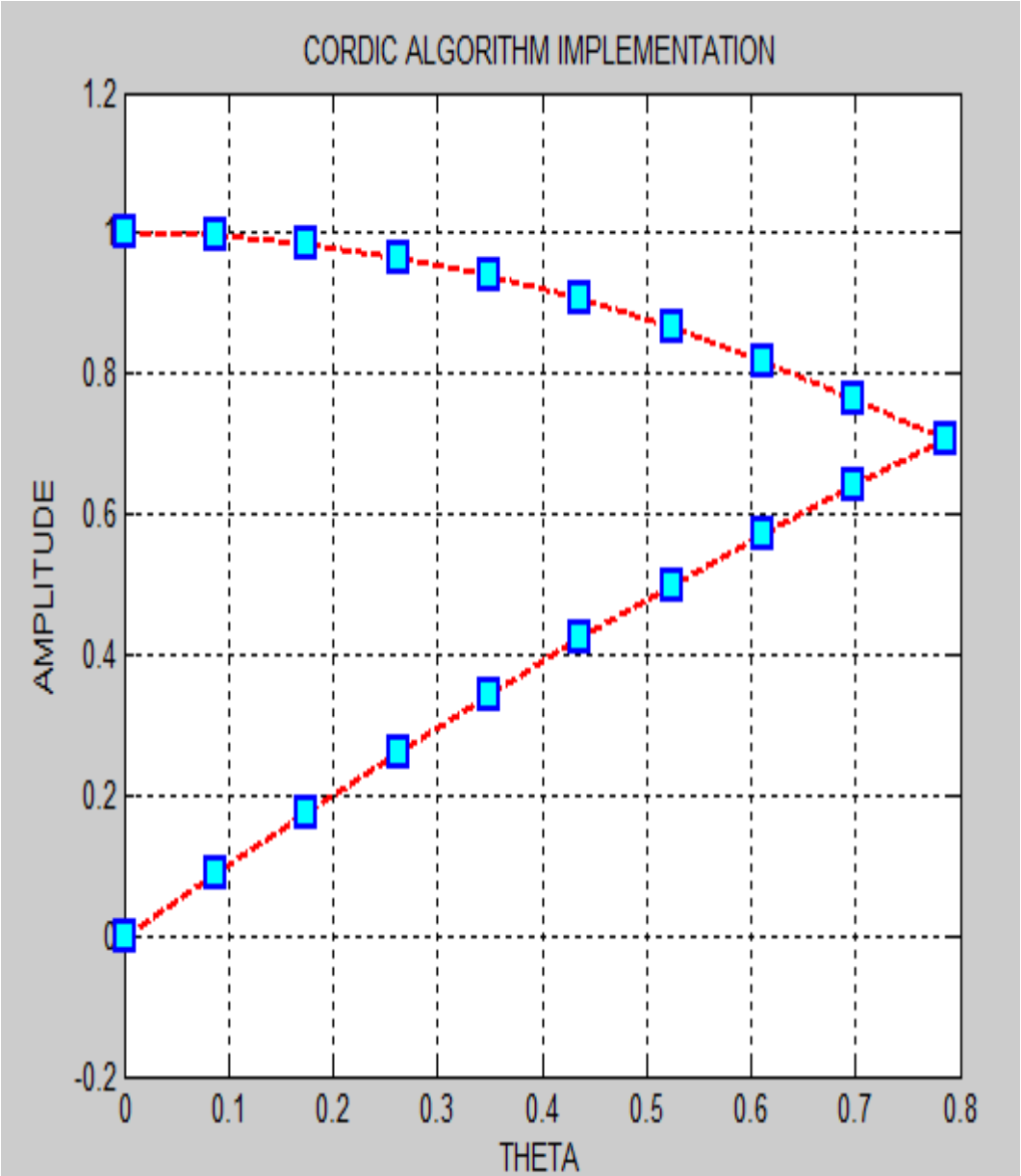


Figure 4.2: - Cordic algorithm implementation of Sine and Cosine from 0 to pi/4

4.4 CORDIC ALGORITHM IMPLEMENTATION OF SINE WAVE

We know if we are having sine and cosine wave from 0 to pi/4 then just because of symmetry property we can construct complete cycle of sine and cosine wave.

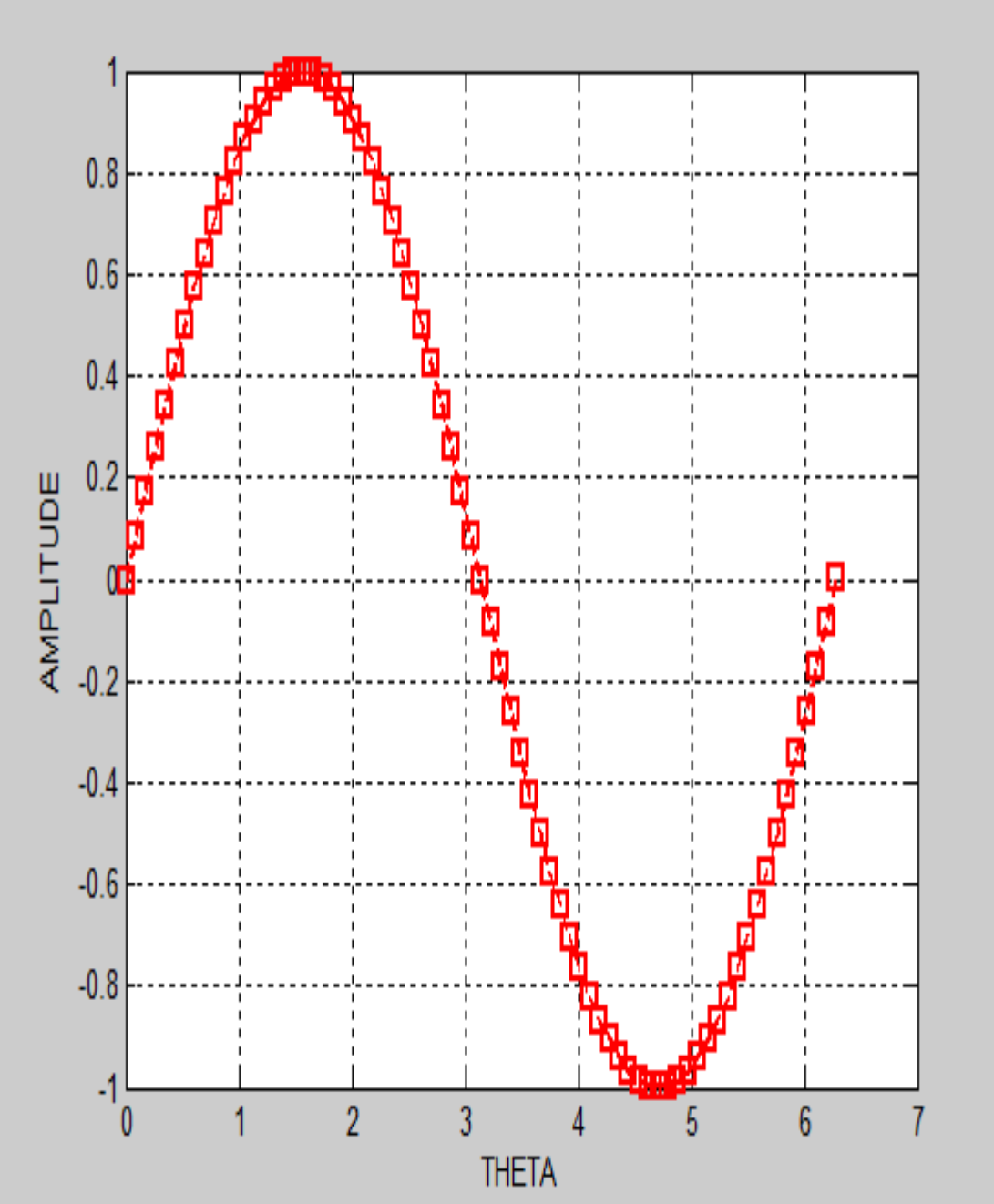


Figure 4.3: - Cordic algorithm implementation of Sine wave

4.5 CORDIC ALGORITHM IMPLEMENTATION OF COSINE WAVE

We know if we are having sine and cosine wave from 0 to pi/4 then just because of symmetry property we can construct complete cycle of sine and cosine wave.

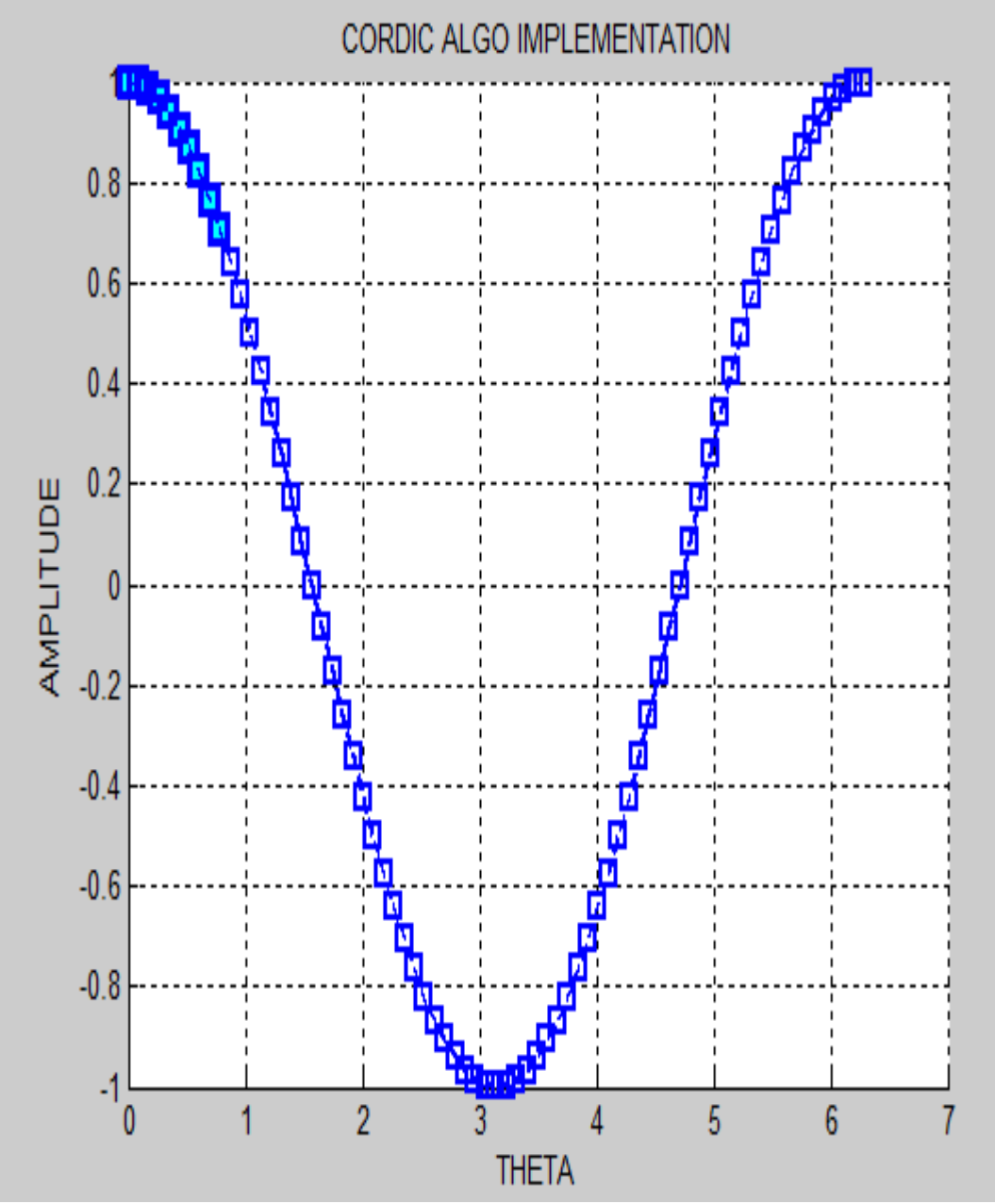


Figure 4.4: - Cordic algorithm implementation of Cosine wave

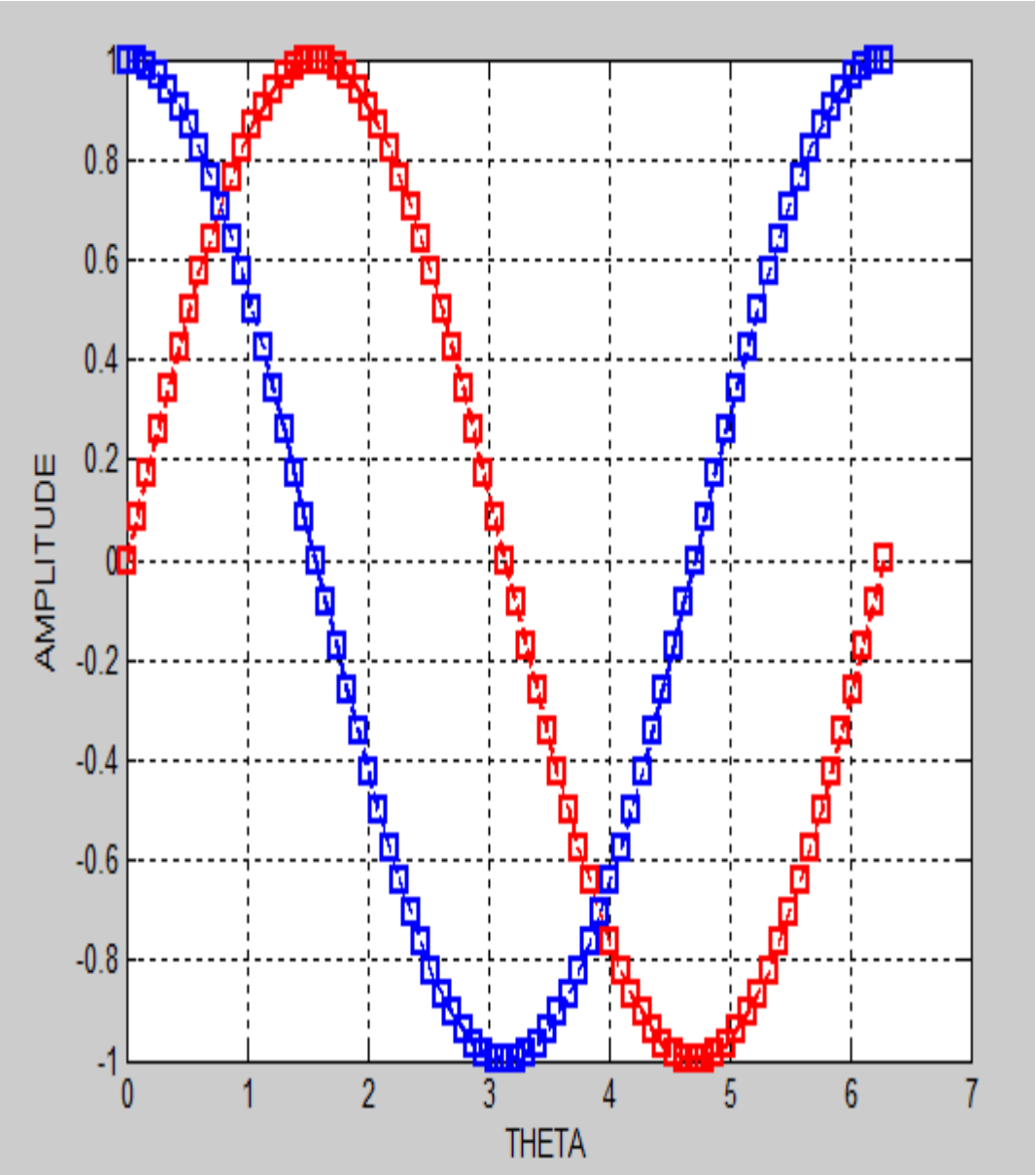


Figure 4.5: - Cordic algorithm implementation of Sine and Cosine wave

4.6 Direct Digital Frequency Synthesizer

DIRECT digital frequency synthesizer (DDFS) is a digital technique for generating sinusoids. Unlike conventional analog oscillator structures, DDFS can be applied where fast frequency switching, fine tuning and a coherent phase relationship among sinusoids are required.

Direct digital synthesis (DDS) is a technique for using digital data processing blocks as a means to generate a frequency- and phase-tunable output signal referenced to a fixed frequency precision clock source. In essence, the reference clock frequency is "divided down" in a DDS architecture by the scaling factor set forth in a programmable binary tuning word. The tuning word is typically 24-48 bits long which enables a DDS implementation to provide superior output frequency tuning resolution.

Today's cost-competitive, high-performance, functionally-integrated, and small package-sized DDS products are fast becoming an alternative to traditional frequency-agile analog synthesizer solutions. The integration of a high-speed, high-performance, D/A converter and DDS architecture onto a single chip (forming what is commonly known as a Complete-DDS solution) enabled this technology to target a wider range of applications and provide, in many cases, an attractive alternative to analog-based PLL synthesizers. For many applications, the DDS solution holds some distinct advantages over the equivalent agile analog frequency synthesizer employing PLL circuitry.There are so many advantages of DDFS.Among those some are:

- Micro-Hertz tuning resolution of the output frequency and sub-degree phase tuning Capability, all under complete digital control.
- Extremely fast "hopping speed" in tuning output frequency (or phase), phase-continuous frequency hops with no over/undershoot or analog-related loop settling time anomalies.

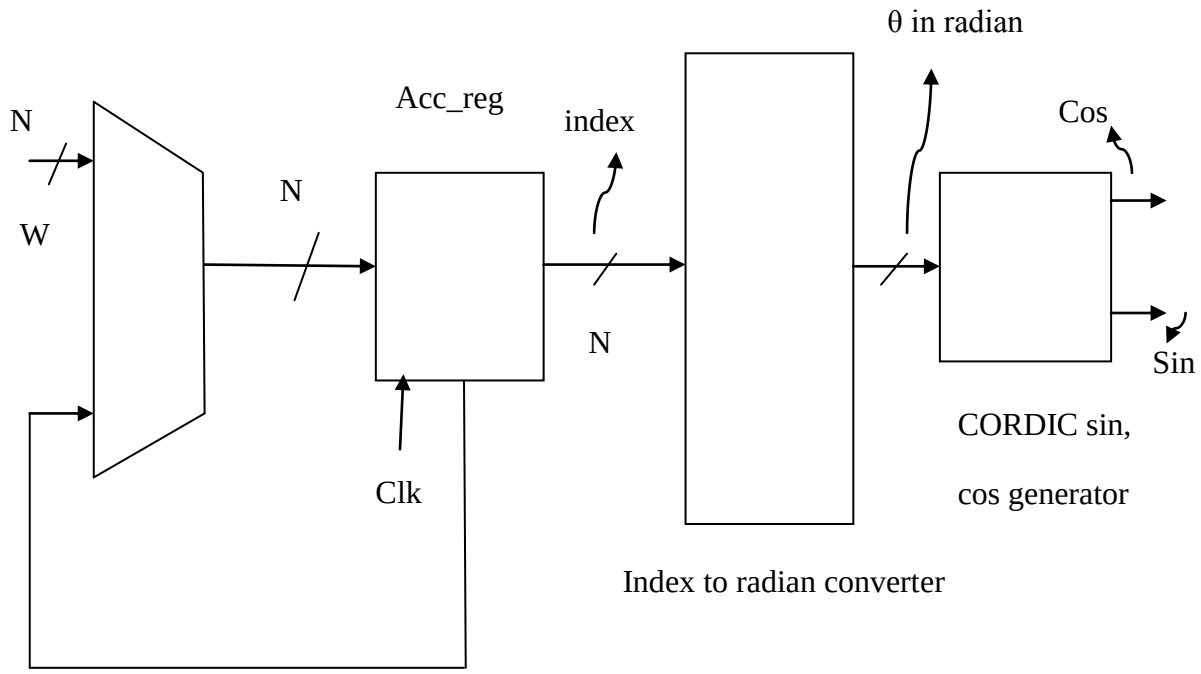
The circular-mode CORDIC (coordinate rotation digital computer) algorithm is an iterative method to compute sine/cosine values, in which an initial vector is rotated by a predetermined sequence of sub-angles in such a way that the summation of the rotations approaches the given angle. CORDIC has been a widely-adopted method for implementing the sine/cosine generator in DDFS . When compared to the ROM look-up-table approach, where all the required sine/cosine sample values are stored in a ROM [5], the CORDIC approach does not lead to the exponential growth of the hardware. CORDIC is a very interesting technique for phase to sine amplitude conversion. This algorithm proposed by J.E.Volder utilizes dynamic transformation rather than ROM static addressing. 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. The trigonometric CORDIC algorithms were originally developed as a digital solution for real-time navigation problems. The original work is credited to Jack Volder. Extensions to the CORDIC theory based on work by John Walther and others provide solutions to a broader class of functions. The CORDIC algorithm has found its way into diverse applications including the microprocessors, calculator, radar signal processors and robotics.

4.7 CORDIC BASED DDFS

The circular-mode CORDIC (coordinate rotation digital computer) algorithm is an iterative method to compute sine/cosine values, in which an initial vector is rotated by a predetermined sequence of sub-angles in such a way that the summation of the rotations approaches the given angle. CORDIC has been a widely-adopted method for implementing the sine/cosine generator in DDFS. When compared to the ROM look-up-table approach, where all the required sine/cosine sample values are stored in a ROM, the CORDIC approach does not lead to the exponential growth of the hardware. The Figure of cordic based DDFS is shown in figure

Figure: 4.6: - DDFS with CORDIC block



4.8 SINE WAVE GENERATION

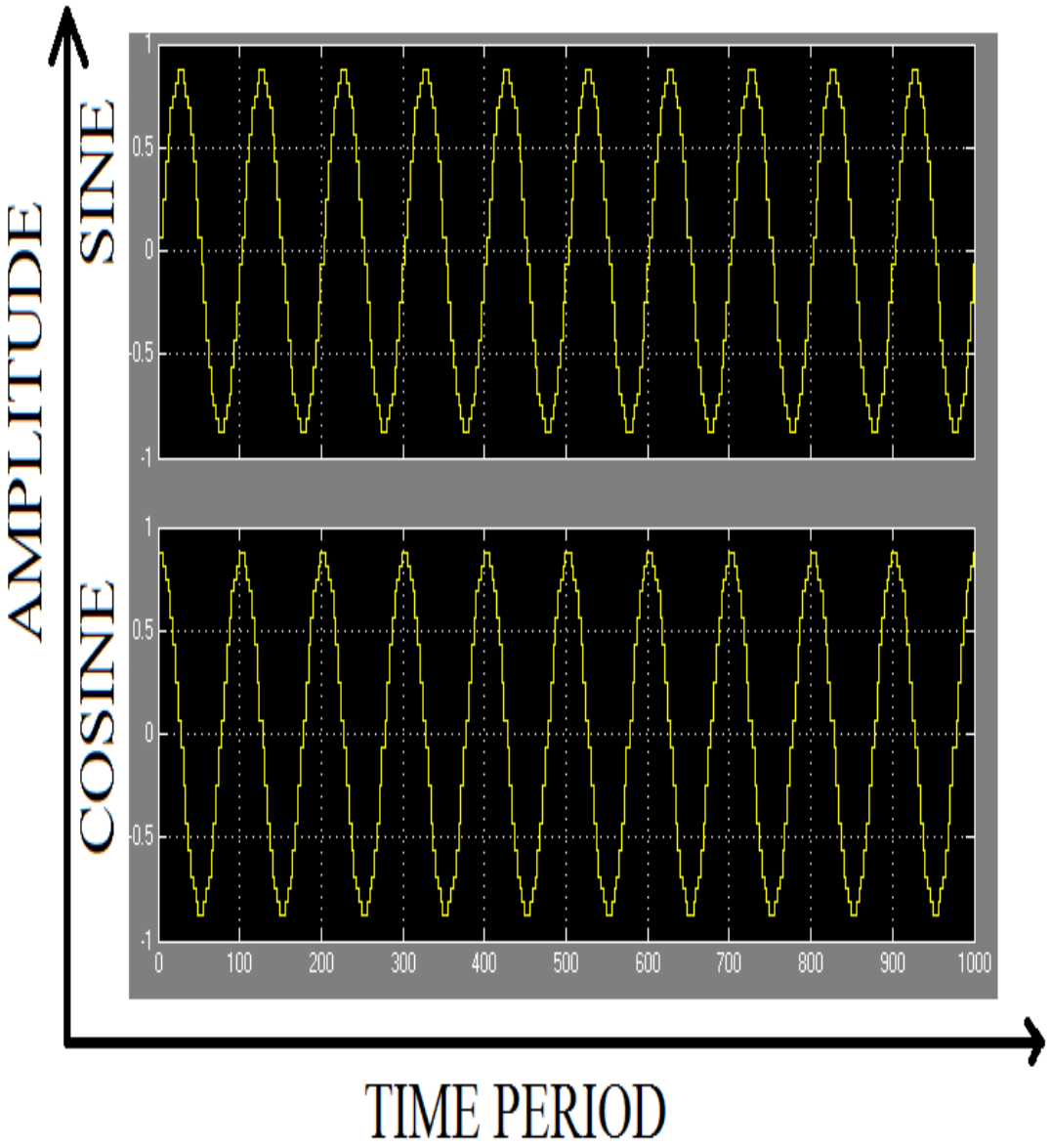


Figure 4.7: - Sine wave generation

4.9 SINE AND COSINE WAVE GENERATION

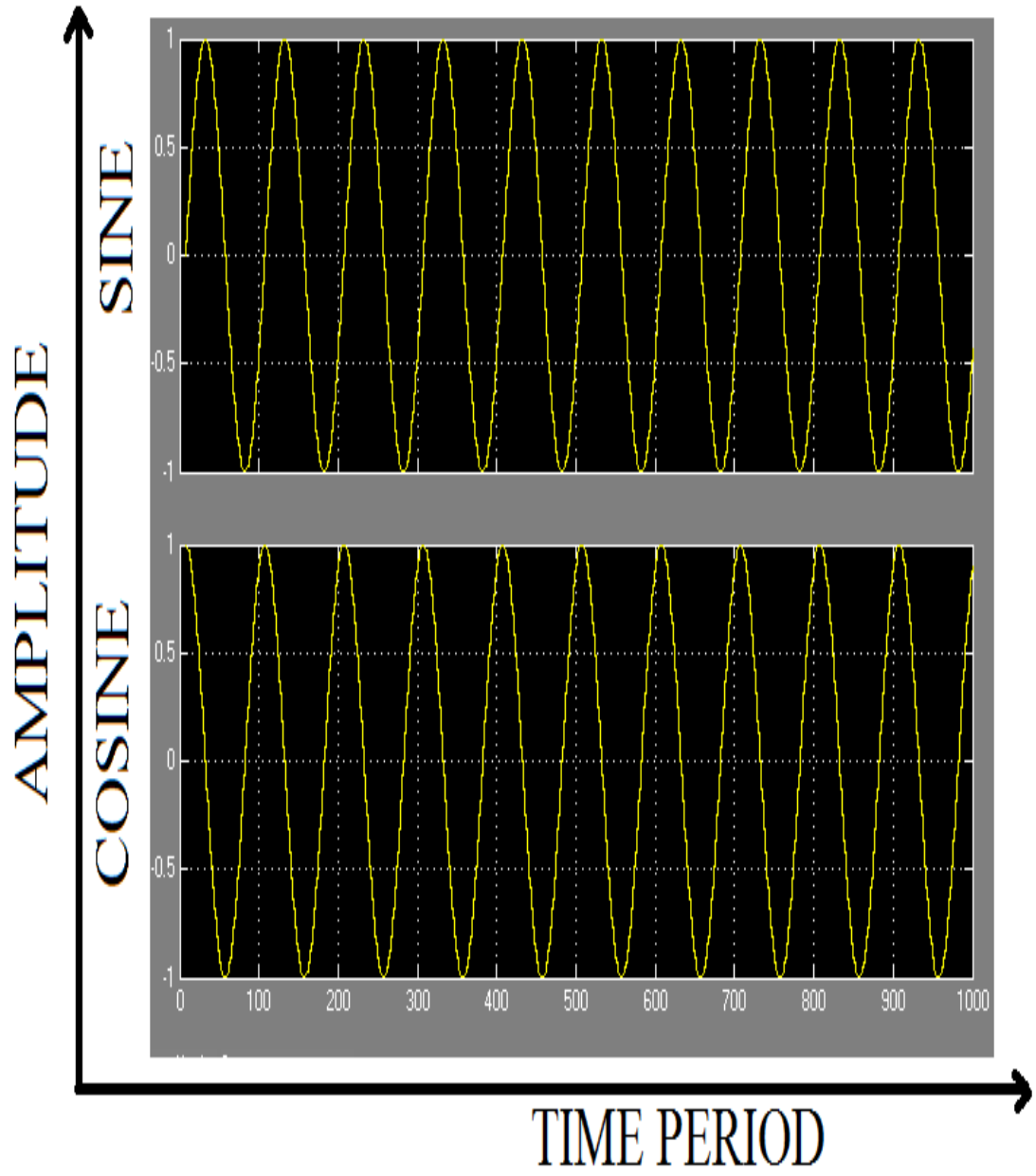


Figure 4.8: - Sine and Cosine wave generation with Phase Dithering effect

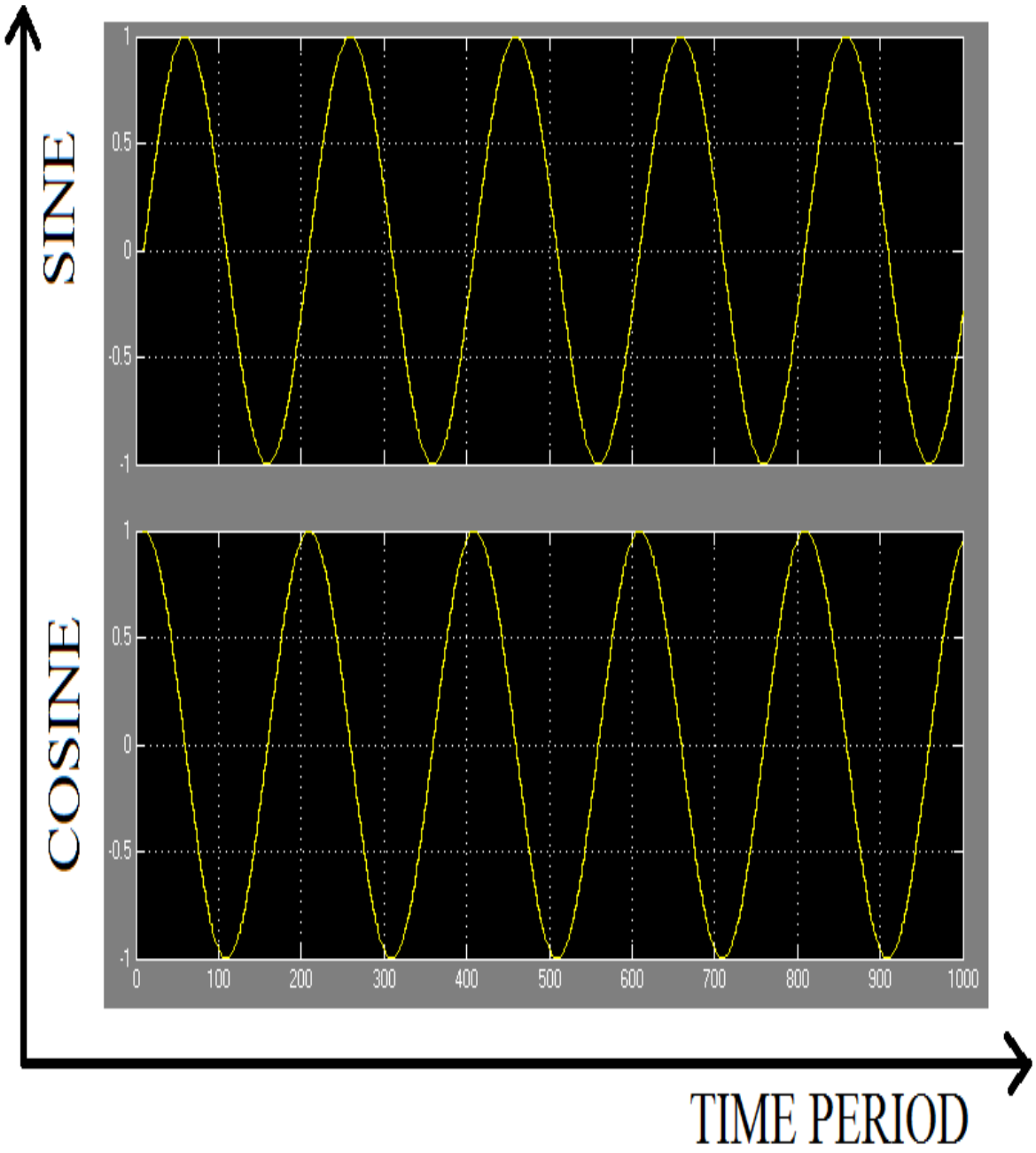


Figure 4.9: - Sine and Cosine wave generation with Phase Dithering effect

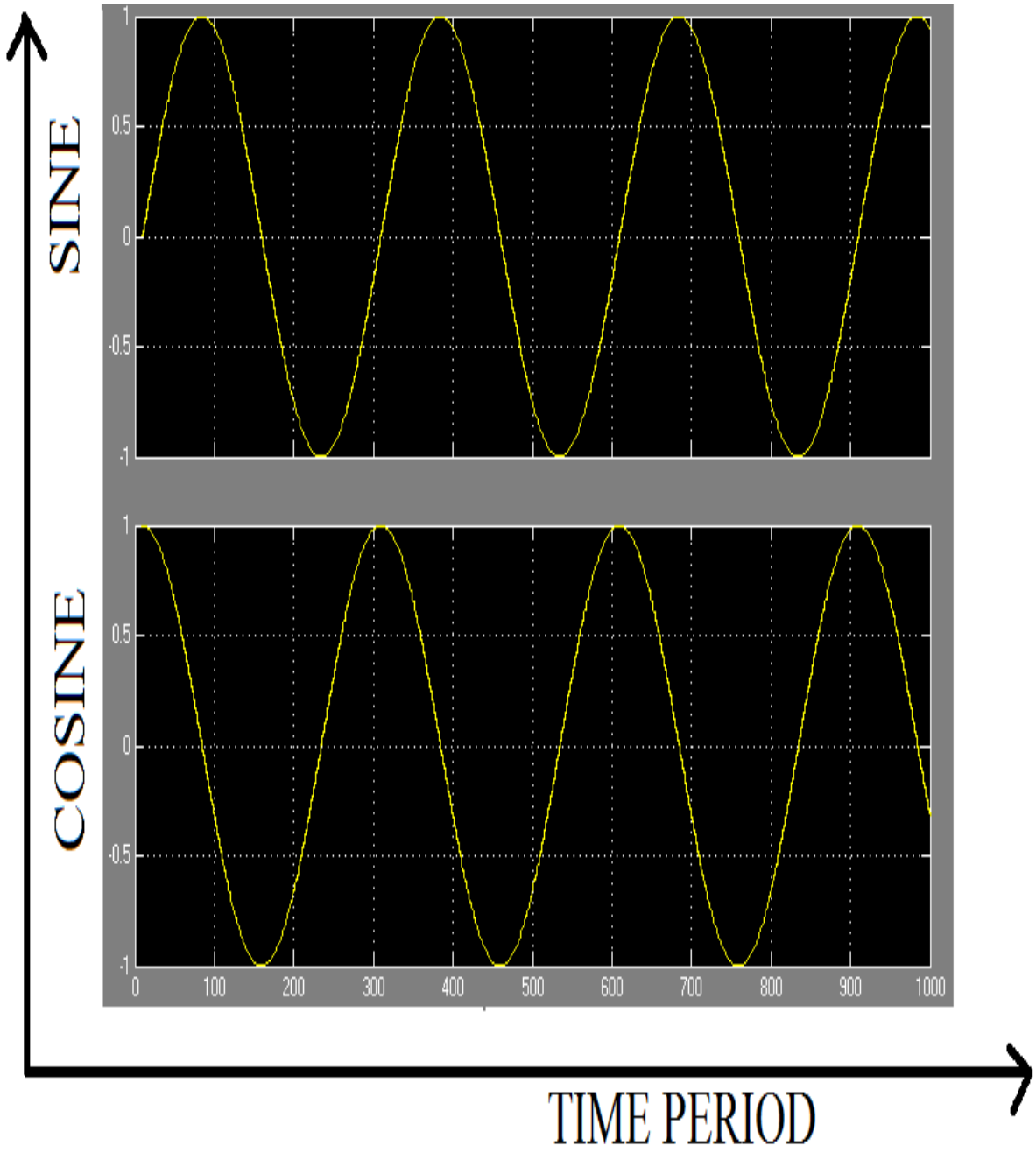


Figure 4.10: - Sine and Cosine wave generation with Phase Dithering effect

CHAPTER 5

CONCLUSION

The aim of this thesis report is to study different types of architecture for DDFS(Direct Digital Frequency Synthesizer) and to know CORDIC(Coordinate Rotation Digital Computer) which is the main part of DDFS. So we have first discussed about ROM based DDFS, The main disadvantage of ROM based DDFS is memory requirement and power dissipation. After this we discussed about DDFS architecture which require less ROM, because we store cycle of sine and cosine only up to 0 - $\pi/4$. We studied about three CORDIC based DDFS architectures and then we came to know pipelined cordic based DDFS has more speed but require more hardware. Folded cordic based DDFS architecture requires less area, but speed of that architecture is less. So there is trade off between area and speed. According to our requirement we can use selected architecture. After implementation of cordic based DDFS we have got following SFDR (Spurious Free Dynamic Range)

Noise Shaping	SFDR	Hardware Parameter
None	96 db	Phase width = 29
Phase Dithering	108 db	Phase width = 29

References

[1]. Shoab Ahmed khan, “Digital Design of Signal Processing Systems” John Wiley & Sons, 2011.

[2]. K. Murota, K. Kinoshita and K. Hirade, “GMSK modulation for digital mobile telephony,” IEEE Transactions on Communications, 1981, vol. 29, pp. 1044 1050.

[3]. J. Volder, “The CORDIC computing technique,” IRE Transactions on Computing, 1959, pp. 330 334.

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

[5]. S. Wang, V. Piuri and E. E. Swartzlander, “Hybrid CORDIC algorithms,” IEEE Transactions on Computing, 1997, vol. 46, pp. 1202 1207.

[6]. T. Rodrigues and J. E. Swartzlander, “Adaptive CORDIC: using parallel angle recoding to accelerate rotations,” IEEE Transactions on Computers, 2010, vol. 59, pp. 522 531.

[7]. P. K. Meher, J. Valls, T. B. Juang, K. Sridharan and K. Maharatna, “50 years of CORDIC: algorithms, architectures, and applications,” IEEE Transactions on Circuits and Systems I, 2009, vol. 56, pp. 1893 1907.

[8]. T. Zaidi, Q. Chaudry and S. A. Khan, “An area and time efficient collapsed modified CORDIC DDFS architecture for high rate digital receivers,” in Proceedings of INMIC, 2004, pp. 677 681.

[9]. D. De Caro, N. Petra and G. M. Strollo, “A 380 MHz direct digital synthesizer/mixer with hybrid CORDIC architecture in 0.25 micron CMOS,” IEEE Journal of Solid State Circuits, 2007, vol. 42, pp. 151 160.

[10]. Davide De caro, Nicola Petra, Antonio Giuseppe Maria Strollo, “Direct Digital Frequency Synthesizer using Nonuniform Piecewise-Linear Approximation,” IEEE Tranactions on Circuits and Systems, 2011, vol.58, pp. 2409 2419.

[11]. A. Madisetti, A. Y. Kwentus, and A. N. Willson Jr., “A 100-MHz, 16-b, direct digital frequency synthesizer with a 100-dBc spurious-free dynamic range,” IEEE J. Solid-State Circuits, Aug. 1999, vol. 34, no. 8, pp. 1034–1043.

[12]. L. S. Chimakurthy, M. Ghosh, F. F. Dai, and R. C. Jaeger, “A novel DDFS using nonlinear ROM addressing with improved compression ratio and quantization noise,” IEEE Trans. Ultrason., Ferroelectr.,Freq. Control, Fab. 2006, vol. 53, no. 2, pp. 274–283.

[13]. J. Vankka, “Methods of mapping from phase to sine amplitude in direct digital synthesis,” in Proc. 1996 IEEE Int. Freq. Contr. Symp., 1996, pp. 942–950.

[14]. J. M. P. Langlois and D. Al-Khalili, “Phase to sinusoid amplitude conversion techniques for direct digital frequency synthesis,” IEE Proc. Circuits Devices Syst., Dec. 2004, vol. 151, no. 6, pp. 519–528.

[15]. D. De Caro and A. G. M. Strollo, “High-performance direct digital frequency synthesizers using piecewise-polynomial approximation,” IEEE Trans. Circuit Syst. I, Reg. Papers, Feb.2005, vol. 52, pp. 324–336.

[16]. Deprettere, E., Dewilde and Udo, R., "Pipelined CORDIC Architecture for Fast VLSI Filtering and Array Processing," Proc. ICASSP'84, 1984, pp. 41.A.6.1- 41. A.6.4.

[17]. C. Y. Kang and E. E. Swartzlander, Jr., “Digit-pipelined direct digital frequency synthesis based on differential CORDIC,” IEEE Trans. Circuits Syst. I, Reg. Papers, May 2006, vol. 53, pp. 1035–1044.

[18]. V. E. Van Duzer, "A 0-50 MHz frequency synthesizer with excellent stability, fast switching and fine resolution," Hewlett-PackardJ., vol.15, May 1964, pp. 1-8.

[19]. E. Renschler and B. Welling, "An integrated circuit phase locked loop digital frequency synthesizer," Motorola Semiconductor Products, Inc., Application Note 463.

[20]. J. Tierney, C.M. Rader and B. Gold, "A digital frequency synthesizer," IEEE Transactions on Audio and Electroacoustics, pp.48-57, vol. AU-19, 1971.

[21]. Qian M , "Application of CORDIC Algorithm to Neural Networks VLSI Design", IMACS Multiconference on Computational Engineering in Systems Applications (CESA), Beijing, China, October 4-6, 2006.

[22]. Lin C. H. and Wu A Y, "Mixed-Scaling-Rotation CORDIC (MSRCORDIC) Algorithm and Architecture for High-PerfonnanceVector Rotational DSP Applications", Volume 52, pp 2385- 2398, November 2005.