TRİGONOMETRİK FONKSİYONLARIN CORDIC ALGORİTMASIYLA ÇEVRİM TABLOLARI KULLANILMADAN HESAPLANMASI VE DONANIMININ UYGULANMASI

COMPUTATION OF THE TRIGONOMETRIC FUNCTIONS WITH CORDIC ALGORITHM WITHOUT USING LOOK-UP TABLES AND ITS IMPLEMENTATION

EMRAH AYTÖRE

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK ve ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2010

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan :….....(İmza).....
(Unvanı, Adı ve Soyadı)

Üye (Danışman) :….....(İmza).....
(Unvanı, Adı ve Soyadı)

Üye (Varsa İkinci Danışman) :….....(İmza).....
(Unvanı, Adı ve Soyadı)

Üye :….....(İmza).....
(Unvanı, Adı ve Soyadı)

Üye :….....(İmza).....
(Unvanı, Adı ve Soyadı)

ONAY

Bu tez ...../...../..... tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

Prof. Dr. Adil DENİZLİ
Fen Bilimleri Enstitüsü Müdürü

TRİGONOMETRİK FONKSİYONLARIN CORDIC ALGORİTMASIYLA ÇEVRİM TABOLOLARI KULLANILMADAN HESAPLANMASI VE DONANIMININ UYGULANMASI

Emrah AYTÖRE

ÖZ

Günümüze kadarki süreçte, sinyal işleme uygulamalarında farklı yöntemler kullanılmıştır. Düşük güç tüketimi, yüksek hız, donanımda kullanılan alan vb.. sistemin verimli çalışmasını etkileyen performans kriterlerini en iyi seviyeye getirme isteği donanım tabanlı sistemlerin bir çok sinyal işleme uygulamasında yaygın olarak kullanılmasına olanak tanımıştır. Donanım tabanlı sistemlerde, uygulama türüne göre farklı algoritma mimarileri geliştirilmiştir. Bu algoritmalardan biri de iteratif öteleme-toplama mantığı ile çalışan CORDIC (COordinate Rotational Digital Computer) algoritmalarıdır. CORDIC algoritması trigonometrik fonksiyonların hesaplanmasında, içersinde HFD (Hızlı Fourier Dönüşümü) operasyonlarının kullanıldığı DSP (Digital Signal Processing) uygulamalarında, çarpım işlemlerinin yoğun kullanıldığı matematiksel işlemlerde ve çeşitli radar uygulamalarında yaygın olarak kullanılan etkili bir algoritmadır. Bu tez çalışmasında trigonometrik fonksiyonların hesaplanmasında Klasik CORDIC algoritmasında ROM (Read Only Memory) kullanımını ortadan kaldıran CORDIC yöntemleri incelenmiştir. Trigonometrik fonksiyonların hesaplanmasında sonuca yüksek doğruluk oranlarında ve düşük iterasyon sayısında ulaşabilen bir CORDIC yöntemi tez çalışmaları kapsamında tasarlanmıştır. Farklı CORDIC yöntemlerinin detaylı performans analizlerini gerçekleyebilen kullanıcı dostu bir grafik arayüze sahip MATLAB (MATrix LABoratuary) yazılım uygulaması gerçeklenmiş olup, Xilinx Spartan 300-E geliştirme kiti üzerinde, farklı CORDIC yöntemlerinin FPGA (Field Programmable Gate Array) uygulamaları gerçekleştirilerek, donanımsal performans karşılaştırmaları yapılmıştır. Ayrıca tez çalışmasında CORDIC algoritmalarının kullanıldığı uygulamalar hakkında bilgiler verilmiştir.

ANAHTAR SÖZCÜKLER: CORDIC, trigonometrik fonksiyonlar, vektör rotasyonları, FPGA.

Danışman: Doç. Dr. Ali Ziya ALKAR, Hacettepe Üniversitesi, Elektrik ve Elektronik Mühendisliği Bölümü.

COMPUTATION OF THE TRIGONOMETRIC FUNCTIONS WITH CORDIC ALGORITHM WITHOUT USING LOOK-UP TABLES AND ITS IMPLEMENTATION

Emrah AYTÖRE

ABSTRACT

In the period up to today, different methods were used in signal processing applications. The need to reach the best performance criterias that affects the efficient system operation such as low power consumption, high speed, the area used for hardware facilitates hardware based systems to be broadly used in many signal processing applications. In hardware based systems according to application types different algorithm architectures are developed. One of these algorithms that works with iterative shift-adding method is the CORDIC (Coordinate Rotational Digital Computer) algorithm. The CORDIC algorithm is a widely used, effective algorithm in calculating trigonometric functions, DSP (Digital Signal Processing) applications using FFT (Fast Fourier Transform) operations, mathematical calculations involving multiplication and various radar applications. In this thesis work CORDIC methods eliminating the use of [1] ROM (Read Only Memory) in Classic CORDIC algorithm for trigonometric functions' calculation are investigated. A CORDIC method which attains the result in high accuracy and low iteration count in trigonometric function calculations is designed in the thesis work and the publication request for the suggested approach is accepted. A software application using MATLAB (MATrix LABoratuary) GUI that examines different CORDIC methods in detail is realized and using Xilinx Spartan 300-E development kit, FPGA (Field Programmable Gate Array) applications of different CORDIC methods are realized and performance comparisons are made. Also, information about applications using CORDIC algorithms is given in the thesis .

KEYWORDS: CORDIC, trigonometric functions, vector rotations, FPGA.

Supervisor: Assoc. Prof. Dr. Ali Ziya ALKAR, Hacettepe University, Department of Electrical and Electronics Engineering.

TEŞEKKÜR

Bu tezin gerçekleştirilmesinde ve tez çalışmamızın bir ürünü olan makalemizin kabul sürecinin başından sonuna kadar, değerli yorum ve önerileri ile bana yardımcı olan tez danışmanım sayın Doç Dr. Ali Ziya ALKAR'a katkılarından dolayı teşekkür ederim.

Tez çalışmalarım süresince bana maddi ve manevi desteklerini esirgemeyen aileme gösterdikleri özveri ve desteklerinden dolayı teşekkürü bir borç bilirim.

İÇİNDEKİLER DİZİNİ

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

AFD	Ayrık Fourier Dönüşümü
CORDIC	COordinate Rotation Digital Computer (Koordinat Rotasyonlu Sayısal Bilgisayar)
CPU	Central Process Unit (Merkezi İşlem Birimi)
DSP	Digital Signal Processing (Sayısal Sinyal İşleme)
FFT	Fast Fourier Transform (Hızlı Fourier Dönüşümü)
FPGA	Field Programmable Gate Array (Alan Programlanabilir Kapı Dizileri)
DEC	Decimal (Onluk)
DFT	Discrete Fourier Transform (Ayrık Fourier Dönüşümü)
HEX	Hexadecimal (Onaltılık)
HARICP	Highly Accurate Reduced Iterations CORDIC Processor (Yüksek Doğruluklu Azaltılmış İterasyonlu CORDIC İşlemcisi)
HFD	Hızlı Fourier Dönüşümü
IP	Intellectual Property (Fikri Mülkiyet)
LSB	Least Significant Bit (En Az Önemli Bit)
LUT	Look Up Table (Çevrim Tablosu)
MATLAB	MATrix LABoratuary (Matris Laboratuarı)
MSB	Most Significant Bit (En Önemli Bit)
PPI	Plan Position Indicator (Plan Mevki Göstericisi)
ROM	Read Only Memory (Salt Okunur Bellek)
RTL	Register Transfer Level (Yazmaç Transfer Seviyesi)
SoC	System on Chip (Tümdevre Üstü Sistem)
SRVÖ	Sayısallaştırılmış Radar Video Örnekleri
VHDL	VHSIC Hardware Description Language (Çok Yüksek Hızlı Tümleşik Devreler Donanım Tanımlama Dili)
VHSIC	Very High Speed Integrated Circuits (Çok Yüksek Hızlı Tümleşik Devreler)

SÖZLÜK DİZİNİ

TÜRKÇE	İNGİLİZCE
Açı Yakınsama Hatası	Angle Approximation Error
Alan	Domain
Alan Programlanabilir Kapı Dizileri	Field Programmable Gate Array
Alt Rotasyon	Sub Rotation
Aşama, Rotasyon adımı	Stage
Ardışık Düzen, Boru Hattı	Pipeline
Arta kalan	Residue
Ayrık Zamanlı Veri İşleme	Discrete Time Data Process
Azaltım Tekniği	Reduction Technique
Birim Çember	Unit Circuit
Çarpıcı	Multiplier
Çevrim Tablosu	Look up table, ROM
Çeyrek	Quadrant
Çok Yüksek Hızlı Tümleşik devreler	Very High Speed Integrated Circuits
Çoğullama Çözücü	Demultiplexer
Dairesel	Circular
Dilim	Slice
Donanım Tanımlama Dili	Hardware Description Language

Donanım Tüketimi	Hardware Consumption
Durum Makinesı Kod Bloğu	State Machine Code Block
En Düşük Bit	Least Significant Bit
En Önemli Bit	Most Significant Bit
Fikri Mülkiyet	Intellectual Property
Güç	Power
Hareket Kestirimi	Motion Prediction
Hız	Speed
Hızlı Fourier Dönüşümü	Fast Fourier Transform
İşaret	Sign
İşlemci	Processor
İterasyon	Iteration
Kartezyen Koordinat Düzlemi	Cartesian Coordinate Plane
Katlama	Folding
Katsayı	Coefficient
Kayan Noktalı Sayılar	Floating Point Numbers
Kayan Yazmaç	Shift Register
Kod Çözücü	Decoder
Konvensiyonel, Klasik	Conventional
Kutupsal	Polar
Onaltılık	Hexadecimal

Onluk	Decimal
Ölçekleme Faktörü	Scaling Factor
Ölçekleme Bağımsız	Scaling Free
Ön	Pre
Ötele	Shift
Ötele-Topla	Shift-Add
Plan Mevki Göstericisi	Plan Position Indicator
Quantalama, Nicemleme Hatası	Quantization Error
Saat Çevrimi	Clock Cycle
Sabit Noktalı	Fixed Point
Sayısal Sinyal İşleme	Digital Signal Processing
Sinkro Çözümleyici	Synchro Resolver
Son	Post
Sözde	Pseudo
Tam Toplayıcı	Full Adder
Tespit Birimi	Detection Unit
Topla	Add
Toplayıcı	Adder
Tümdevre Üstü Sistem	System on Chip
Uçucu Olmayan Bellek, Kalıcı Bellek	Nonvolatile Memory
Üretilen iş, iş çıkarma yeteneği	Throughput
Vektör rotasyonu (dönüşü)	Vector Rotation

Yeni Gerçekte Ölçekleme Bağımsız	New Virtually Scaling Free
Yol Seçici	Multiplexer
Yönerge Gecikmesi	Latency
Yuvarlama Hatası	Rounding Error

1. GİRİŞ

Bu bölümde öncelikle tez çalışmasının amacı hakkında bilgiler verilecektir. CORDIC (COordinate Rotational Digital Computer) algoritmalarının trigonometrik fonksiyonların hesaplanmasında ne gibi faydaları olduğu, algoritmada çevrim tablosu (LUT) kullanımının dezavantajları, çevrim tablosu kullanımını ortadan kaldıran CORDIC mimarileri ve getirmiş olduğu avantajlar anlatılacaktır. Ayrıca farklı CORDIC mimarileri için uygulama sonuçları performans kriterleri açısından karşılaştırılacak olup, algoritmanın uygulama alanları ile ilgili çalışma konularından bahsedilecektir.

1.1. Tezin Amacı

Bu çalışmada trigonometrik fonksiyonların hesaplanmasında donanımı verimli kullanan, sistem performansını artıran CORDIC tekniği [1] ve bu algoritmanın performansını artırmak için günümüze kadar geliştirilmiş yöntemlerin [9], [10], [11], [12] ayrıntılı olarak tanıtılması ve uygulamada sağladıkları avantajların anlatılması amaçlanmıştır. Çevrim tablosu kullanılmaksızın CORDIC algoritmalarının nasıl gerçekleştirilebileceği üzerine kapsamlı bir bilgi kaynağı oluşturulması tezin amaçları arasındadır.

Bu tez çalışması kapsamında farklı CORDIC algoritmalarının yazılımsal ve donanımsal uygulama sonuçları performans kriterleri açısından karşılaştırılmıştır.

1.2. Tezin Hedefi

Bu çalışmada; donanım çarpıcısı bulunmayan programlanabilir işlemciler üzerinde uygulanabilir trigonometrik fonksiyon hesaplama yöntemleri arasında donanımı verimli kullanan, hızlı ve düşük maliyetli CORDIC algoritması üzerinde durulmuştur.

Ayrıca bu çalışmada; iyileştirilmiş CORDIC algoritmaları arasında sistem performans kriterlerini oldukça verimli kullanmaya olanak tanıyan çevrim tablosu kullanılmaksızın CORDIC algoritmalarının gerçekleştirilmesi konusunda günümüze kadar ortaya atılmış CORDIC yaklaşımlarının bir arada sunulduğu bir bilgi kaynağı oluşturulması hedeflenmiştir.

1.3. Tez Planı

Bu tez, trigonometrik fonksiyonların hesaplanmasında çevrim tablosu kullanmayan CORDIC algoritmalarının incelenmesini ve performans karşılaştırmalarının yapılmasını içeren bir çalışmadır.

Tez sekiz bölüm ve eklerden oluşmaktadır. Birinci bölüm tez çalışmasının amacı hakkında genel bilgiler vermektedir. Bu bölümde ayrıca CORDIC algoritmalarının önemi ve uygulama alanları üzerinde durulmuştur.

İkinci bölümde ise tezin ana teması hakkında bilgi sahibi olunabilmesi için Klasik CORDIC algoritması [1] hakkında teorik bilgiler açıklanmıştır.

Üçüncü bölümde trigonometrik fonksiyonların hesaplanması hakkında teorik bilgiler verilmiştir. Tez çalışması dairesel CORDIC rotasyonları ile sınırlı olduğu için üçüncü bölümde sadece sinüs ve kosinüs fonksiyonlarının hesaplanmasına ilişkin bilgiler verilmiş olup, sinüs ve kosinüs fonksiyonlarının hesaplanmasında kullanılan farklı yöntemler gelişim sırasına göre açıklanmıştır.

Dördüncü bölümde trigonometrik fonksiyonların hesaplanmasında çevrim tablosu kullanan Klasik CORDIC algoritması [1] ve çevrim tablosu kullanmayan farklı CORDIC algoritmaları [9], [10], [11], [12] detaylı bir şekilde incelenmiştir.

Beşinci bölümde, tez çalışması kapsamında tasarlanmış olan “GUICORDIC” MATLAB programı kullanıcı grafik arayüzü tanıtılmış olup, bu program kullanılarak, dokümanda adı geçen farklı CORDIC algoritmalarının analizleri yapılmıştır. Bu bölümde MATLAB programının ürettiği analiz sonuçları verilmiş ve performans karşılaştırmaları yapılmıştır. Ayrıca bu bölümde bahsi geçen CORDIC algoritmalarının FPGA uygulaması, FPGA geliştirme kiti kullanılarak gerçekleştirilmiş olup, Xilinx ISE programı, geliştirme kiti ile birlikte gerçek zamanlı kullanılarak elde edilen sentez raporlarına dokümanda yer verilmiştir. Farklı CORDIC algoritmaları için elde edilen FPGA sentez sonuçlarının donanımsal performans karşılaştırması bu bölümde yapılmıştır.

Altıncı bölümde ise CORDIC algoritmalarının kullanıldığı birkaç önemli uygulama konusu üzerinde durulmuştur.

Yedinci bölümde yapılan analizlerin sonuçları değerlendirilmiştir. Son bölümde ise daha ileri çalışmalar için öneriler sunulmuştur.

Bu tez çalışmasında analizleri gerçeklenen CORDIC uygulamalarının simülasyon ortamındaki yazılımsal uygulaması MATLAB, algoritmaların donanımsal uygulaması ise Xilinx firmasının Spartan 3-E FPGA geliştirme kiti ve Xilinx-ISE programı kullanılarak, VHSIC donanım tanımlama dili (VHDL) temelli gerçeklenmiştir.

2. CORDIC ALGORİTMASI

2.1. Klasik CORDIC Algoritması

CORDIC, “COordinate Rotational DIgital Computer” söz diziminin ilk harflerinden oluşmaktadır. CORDIC, hafıza ve merkezi işlem birimi (CPU) sınırlanmış gömülü sistemlerde yaygın olarak kullanılmaktadır. Trigonometrik fonksiyonları hesaplamada basit ve hızlı yakınsama özelliğine sahip oluşu algoritmayı diğer yaklaşımlara üstün kılmaktadır.

CORDIC algoritması vektör rotasyonları sadece öteleme ve toplama operasyonları ile yürütülebilmektedir. Bu sebeple, çok sayıda kapı kullanımı gerektiren ve daha maliyetli olan donanım çarpıcıları yerine CORDIC algoritmalarının kullanımı tasarımcılar tarafından tercih edilmektedir. Günümüzde trigonometrik fonksiyon hesapları ve kartezyen koordinatlar ile polar koordinatlar arasındaki dönüşüm operasyonlarını içeren çok sayıda uygulamada CORDIC kullanılmaktadır.

İlk olarak 1959 yılında J. E. Volder [1] tarafından ortaya konulan CORDIC algoritması; kartezyen koordinat sisteminde bir vektörün döndürülerek vektöre ait açı, uzunluk ve yeni kartezyen koordinat bileşenlerinin hesaplanması esasına dayanan donanımı hızlı ve etkin kullanan bir algoritmadır.

Algoritma daha sonra Walther [2] tarafından genişletilerek çarpma, bölme, karekök hesabı ve hiperbolik fonksiyonları hesaplayabilecek şekilde geliştirilmiştir.

CORDIC algoritması; trigonometrik, eksponansiyel, logaritmik vb.. fonksiyonların çözümünde;

- iteratif çözümleme mantığına sahip,
- donanımı verimli kullanan,
- sadece öteleme - toplama (shift-add) operasyonlarına sahip
- küçük bir çevrim tablosuna (look-up table/ROM(Read Only Memory)) ihtiyaç duyan

son derece etkili ve hızlı bir algoritmadır.

Klasik CORDIC algoritması [1] her iki yönde küçük vektör rotasyonları ile yürütülür. Bu vektör rotasyonlarında kullanılan her bir açı "mikro rotasyon açısı" (α) olarak bilinmektedir. CORDIC algoritmasında hedef θ (hedef açısı) açısına ulaşabilmek için, her bir iterasyonda önceden tanımlanmış bir α_i (ilgili iterasyondaki mikro-rotasyon açısı) mikro rotasyon açısı kadar vektör döndürülür. α_i mikro rotasyon açı değerleri küçük bir çevrim tablosunda (look-up table/ROM) tutulur. Algoritma giriş verilerinin n-bit uzunluğunda olduğu varsayılırsa, Klasik CORDIC algoritması [1] iterasyon sayısı n oluncaya kadar yürütülür, ardından sonuçlar elde edilir.

2.1.1. Klasik CORDIC Algoritması Standart Rotasyon Denklemlerinin Elde Edilmesi

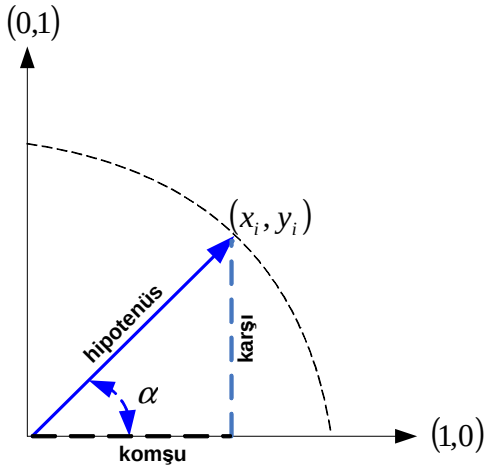
Farklı mimarilere sahip çok sayıda CORDIC algoritması, Klasik CORDIC algoritmasından [1] türemiştir. Bu bölümde ileriki bölümlerde açıklanmış olan CORDIC mimarilerinin anlaşılabilmesi için Klasik CORDIC algoritması [1] hakkında temel bilgiler verilmiş olup, standart CORDIC rotasyon denklemlerinin nasıl elde edildiği adım adım açıklanmıştır.

2.1.1.1. Birim Çember Mantığı

Birim vektör trigonometrik veya dairesel fonksiyonların tanımlanmasında kullanılır. Bir vektör genlik ve yön ile temsil edilir. Eğer vektörün genliği birim uzunluğa eşit ise "birim vektör" olarak bilinir.

- Tüm pozitif açılar birim vektör saat yönü tersine döndürülerek bulunur.
- Tüm negatif açılar birim vektör saat yönünde döndürülerek bulunur.

Her iki yönde tam bir tur dönüş için 2π kadarlık bir dönüş (rotasyon) gereklidir.



Şekil 2 - 1. Birim Çember

Birim vektör ve birim çember kullanılarak genel rotasyon denklemleri geliştirilebilir. Şekil 2-1'de α rotasyonuna ve (x_i, y_i) koordinat bileşenlerine sahip bir birim vektör gösterilmektedir.

Birim vektörün birim çember ile kesişim noktası olan uç noktasının koordinatları $\cos(\alpha)$ ve $\sin(\alpha)$ ile açıklanabilir. Şekil 2-1'deki üçgene ait hipotenüs kenarı aynı zamanda birim vektör olduğu için genlik değeri '1' dir. Birim vektör, birim çemberin herhangi bir quadrantında bulunabilir.

Birim vektör, birim çember üzerinde α açısı kadar döndürülecek olursa birim çember üzerinde yeni bir vektör elde edilir. Döndürülmüş vektörün yeni koordinatları (2.1)-(2.4) denklemleri kullanılarak hesaplanır.

$$\cos(\alpha) = \frac{\text{komsu}}{\text{hipotenus}} = \frac{\text{komsu}}{1} = \text{komsu} \tag{2.1}$$

$$\sin(\alpha) = \frac{\text{karsi}}{\text{hipotenus}} = \frac{\text{karsi}}{1} = \text{karsi} \tag{2.2}$$

$$\cos(\alpha) = \frac{x_i}{1} = x_i \tag{2.3}$$

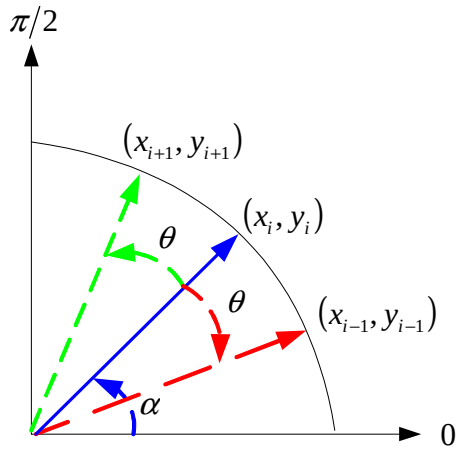
$$\sin(\alpha) = \frac{y_i}{1} = y_i \tag{2.4}$$

Birim vektör pozitif yönde döndürülürse yeni birim vektörün x ekseni ile yaptığı açı $\alpha + \alpha_i$ kadar olur. Birim vektör α açısına sahip olduğu konumdan negatif yönde döndürülürse yeni birim vektörün x ekseni ile yaptığı açı $\alpha - \alpha_i$ kadar olur. (2.5), (2.6) nolu denklemlerde ve Şekil 2-2'de yukarıda ifade edilen durumlar gösterilmiştir.

$$\cos(\alpha \pm \alpha_i) = x_{i+1} \tag{2.5}$$

$$\sin(\alpha \pm \alpha_i) = y_{i+1} \tag{2.6}$$

Yukarıdaki denklemlerdeki '±' her iki yöndeki vektör dönüşünü ifade etmektedir.



Şekil 2 - 2. Birim çember üzerinde iki yönlü vektör rotasyonu

(2.5) ve (2.6) nolu denklemler kosinüs ve sinüs formülleri kullanılarak (2.7) ve (2.8) nolu denklemlere sırasıyla dönüştürülebilir.

$$x_{i+1} = \cos(\alpha \pm \alpha_i) = \cos(\alpha).\cos(\alpha_i) \mp \sin(\alpha).\sin(\alpha_i) \tag{2.7}$$

$$y_{i+1} = \sin(\alpha \pm \alpha_i) = \sin(\alpha).\cos(\alpha_i) \pm \cos(\alpha).\sin(\alpha_i) \tag{2.8}$$

(2.3) ve (2.4) nolu denklemler (2.7) ve (2.8) nolu denklemlerde yerine konulursa sırasıyla (2.9) ve (2.10) nolu denklemler elde edilir.

$$x_{i+1} = \cos(\alpha_i).x_i \mp \sin(\alpha_i).y_i \tag{2.9}$$

$$y_{i+1} = \cos(\alpha_i).y_i \pm \sin(\alpha_i).x_i \tag{2.10}$$

(2.9) ve (2.10) denklemler bir matris çarpımı olarak yeniden düzenlenecek olursa standart CORDIC rotasyonel denklemi (2.11) elde edilir. Verilen bir (x_i, y_i) başlangıç vektörü tek bir iterasyon için yürütülecek olursa (x_{i+1}, y_{i+1}) vektör bileşenleri (2.11) nolu denklem kullanılarak hesaplanabilir.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\alpha_i & \mp\sin\alpha_i \\ \pm\sin\alpha_i & \cos\alpha_i \end{bmatrix} . \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{2.11}$$

n iterasyon için (2.11) nolu rotasyonel denklem yeniden yazılacak olursa:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos(\alpha_{n-1}) & \mp\sin(\alpha_{n-1}) \\ \pm\sin(\alpha_{n-1}) & \cos(\alpha_{n-1}) \end{bmatrix} ..... \begin{bmatrix} \cos(\alpha_0) & \mp\sin(\alpha_0) \\ \pm\sin(\alpha_0) & \cos(\alpha_0) \end{bmatrix} . \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2.12}$$

elde edilir.

Klasik CORDIC algoritması [1] her iki açı yönünde vektör rotasyonları ile yürütülür. Vektör rotasyonunda kullanılan her bir açı mikro rotasyon açısı α_i olarak adlandırılır. θ hedef açısına ulaşabilmek için, her bir iterasyonda bir α_i mikro rotasyon açısı kadar vektör döndürülür.

Denklem (2.12) bir problemi ortaya çıkarmaktadır. Denklem incelendiğinde her alt rotasyon 4 adet çarpma ve 2 adet toplama/çıkarma operasyonuna ihtiyaç duymaktadır. Çarpma işlemi sayısının yüksek oluşu donanımda kullanılan alanı ve işlem zamanını arttıracaktır. Bu problemi ortadan kaldırmak için Denklem (2.12)'de her bir matristeki kosinüs bileşenleri matrisin dışına çarpan olarak alınırsa, CORDIC rotasyonel denklemi (2.13)'deki gibi olacaktır.

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \cos(\alpha_{n-1})...\cos(\alpha_1).\cos(\alpha_0).\begin{bmatrix} 1 & \mp\tan(\alpha_{n-1}) \\ \pm\tan(\alpha_{n-1}) & 1 \end{bmatrix} ..... \begin{bmatrix} 1 & \mp\tan(\alpha_0) \\ \pm\tan(\alpha_0) & 1 \end{bmatrix} . \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2.13}$$

Böylece her alt rotasyon (sub-rotation) için gerekli işlem sayısı azaltılmış olur. Her bir alt rotasyon için çarpma işlemi sayısı 4'ten 2'ye indirgenmiş, toplama işlemi sayısında ise bir değişiklik olmamıştır. Bu proses iteratif olarak çalışmaktadır.

Algoritma giriş verilerinin n-bit uzunluğunda olduğu varsayılırsa, CORDIC algoritması iterasyon sayısı n oluncaya kadar yürütülecek, ardından sonuç bileşenleri elde edilecektir. Bu sebeple algoritmanın iterasyon sayısının belirlenmesinde giriş verilerinin bit uzunlukları büyük önem taşımaktadır.

Denklem (2.14), Denklem (2.13)'ün tüm rotasyon dönüşleri için en genel ifadesidir.

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \cos(\alpha_{n-1})...\cos(\alpha_1).\cos(\alpha_0).\begin{bmatrix} 1 & m.r_{n-1}.\tan(\alpha_{n-1}) \\ -m.r_{n-1}.\tan(\alpha_{n-1}) & 1 \end{bmatrix}\cdots$$

$$\cdots\begin{bmatrix} 1 & m.r_0.\tan(\alpha_0) \\ -m.r_0.\tan(\alpha_0) & 1 \end{bmatrix}.\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \qquad (2.14)$$

$$r_i = Sign\left[\theta - \sum_{j=0}^{i-1} \alpha_j\right]$$

Denklem (2.14)'de 'm' rotasyon tipi seçim biti, 'r_i' her vektör rotasyonunun yönünü temsil etmektedir.

Klasik CORDIC algoritmasında [1], $r_i = 1$ veya -1 (saat yönü veya saat yönü tersi) değerlerinden birini alabilir. θ hedef açısı n adet α_i mikro rotasyon açısı ile temsil edilebilir.

$$\theta = \sum_{i=0}^{n-1} \mathrm{r_i}.\alpha_i \qquad (2.15)$$

Denklem (2.15)'de tanımlandığı gibi her iki yöndeki mikro rotasyon açılarının toplam değeri θ hedef açısını verecektir.

Küçük mikro rotasyonları için $\tan(\alpha_i)$, 2^{-i} kuvvetleri şeklinde temsil edilebilir.

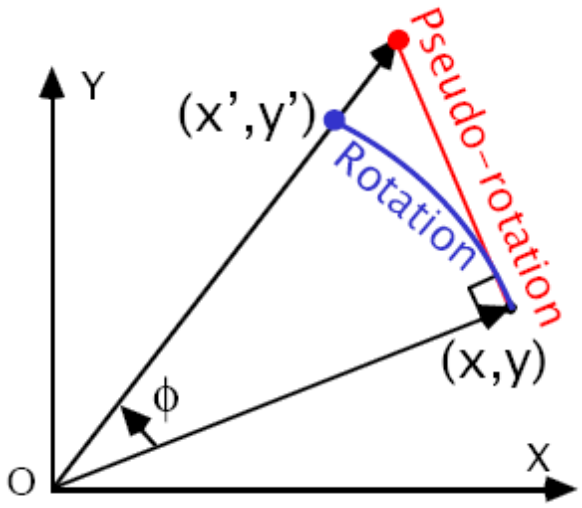
$$\tan(\alpha_i) = 2^{-i} \qquad (2.16)$$

Bir çarpma işleminin hızlı ve kolay yürütülebilmesi için en uygun yol çarpma işleminin 2'nin kuvvetleri (powers of 2) biçiminde Denklem (2.16)'da olduğu gibi ifade edilmesidir. 2'nin kuvveti biçiminde çarpma işlemleri donanımda öteleme işlemlerine karşılık gelmektedir.

Eğer Denklem (2.16)'da eşitliğin her iki tarafının da arktanjantı alınır ve α_i değişkeni Denklem (2.15)'de yerine konursa (2.17) nolu denklemde ifade edilen θ eşitliği elde edilir.

$$\theta = \sum_{i=0}^{n-1} r_i . \tan^{-1}(2^{-i}) \qquad (2.17)$$

Klasik CORDIC algoritmasında [1] her mikro rotasyonda, döndürülmüş vektörün uzunluğu K_i ifadesi ile ölçeklenerek elde edilir. Bu mikro rotasyonlar sözde vektör rotasyonları (pseudo vector rotations) olarak bilinir. Çünkü döndürülmüş vektörün uzunluğu başlangıç vektörü uzunluğu ile aynı değildir. Bu durum Şekil 2-3'de gösterilmiştir.



Şekil 2 - 3. Sözde vektör rotasyonu

Klasik CORDIC algoritmasının [1] iterasyon sayısı K ölçekleme faktörüne bağlıdır. K ölçek faktörü Denklem (2.18)'de ifade edildiği gibi n iterasyon sonrasında sabit bir değere yakınsar. Bu sebeple iteratif algoritmanın en son adımında üretilen koordinat bileşenleri ile Denklem (2.13)'de ifade edilen iterasyon sayısına bağımlı olan sabit ölçek sabiti bir kez çarpılır.

$$K = \prod_{i=0}^{n-1} \cos(\tan^{-1}(2^{-i})) = K_{n-1}.....K_0 = \coprod_{i=0}^{n-1} K_i \cong 0.60725 \qquad (2.18)$$

İterasyon sayısı arttıkça, K ölçekleme faktörü ilave toplama-öteleme operasyonlarına ihtiyaç duyacaktır. Bu durum sistemin verimliliğini etkileyecektir. Burada dikkat edilmesi gereken nokta Klasik CORDIC algoritmasında [1] K ölçek faktörünün iterasyon sayısına bağımlı oluşudur.

Denklem (2.14)'de tanımlı kosinüs ifadeleri yerine, (2.18) nolu denklemde tanımlı K ölçekleme faktörü ifadesi dairesel rotasyon modunda (m=1 için) yerleştirilir ve Denklem (2.16) göz önüne alınırsa; (x_0, y_0) başlangıç vektörüne rotasyonlar uygulandığında n iterasyon adımı için genel CORDIC denklemi (2.19) elde edilecektir.

Çok sayıda dairesel CORDIC algoritması (2.19) nolu denklemden türetilmektedir. İleriki bölümlerde bu algoritmalardan önemli birkaçı detaylı bir şekilde anlatılacaktır.

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = K.\begin{bmatrix} 1 & r_{n-1}.2^{-(n-1)} \\ -r_{n-1}.2^{-(n-1)} & 1 \end{bmatrix}...\begin{bmatrix} 1 & r_1.2^{-1} \\ -r_1.2^{-1} & 1 \end{bmatrix}\begin{bmatrix} 1 & r_0 \\ -r_0. & 1 \end{bmatrix}.\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \qquad (2.19)$$

$$z_{i+1} = z_i - \mathrm{r_i}.\alpha_i$$

$\alpha_i = \tan^{-1}(2^{-i})$ değerleri küçük bir çevrim tablosunda (look-up table/ROM) tutulmaktadır.

Denklem (2.19)'dan görüldüğü gibi CORDIC vektör rotasyonlarında z residue açısı azaltım tekniği (z reduction technique) kullanılır. Her bir iterasyon sonrasında çevrim tablosundan okunan α_i mikro rotasyon açı değeriyle, o an üretilen z_i residue değeri karşılaştırılmakta ve karşılaştırma sonrası yeni residue değeri uygun koşul sağlandığında $z_{i+1} = z_i - \mathrm{r_i}.\alpha_i$ eşitliği göz önüne alınarak hesaplanmaktadır. Hesaplanan residue değeri referans değerinden düşük olduğu zaman algoritma sonlandırılır. Algoritma sonlanıncaya kadar kullanılan alt rotasyon sayısı bize toplam iterasyon sayısını verir.

Kullanılan iterasyon sayısı düşük ise algoritma hızlı, donanım az alan kaplıyor demektir. Aksi durumda algoritma yavaş ve donanım maliyetli olacaktır. Algoritmaya ait akış diyagramı ve detaylı anlatım ileriki bölümlerde açıklanmıştır.

Not: Donanım uygulaması açısından bakıldığında vektör rotasyonu diye bir durum söz konusu değildir. Sadece ardışık ötele ve topla (shift and add) operasyonları söz konusudur.

2.2. CORDIC Algoritmasının Uygulama Alanları

CORDIC yöntemi Fourier dönüşümü [3,4], matris dönüşümleri ve operasyonları [5,6], bilgisayar grafik operasyonları [7], householder dönüşümler [8], robotik uygulamalar, sinyal işleme vb.. çok sayıda uygulama alanında kullanılmaktadır.

2.3. CORDIC Algoritması Modları

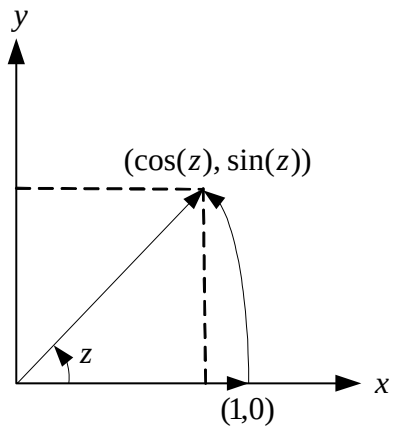
Tüm CORDIC algoritmaları uygulamaya bağlı olarak rotasyonel modu (rotational mode) ve vektörel modu (vectoring mode) rotasyonlar olmak üzere iki çalışma modunda yürütür.

2.3.1. Rotasyonel Modu

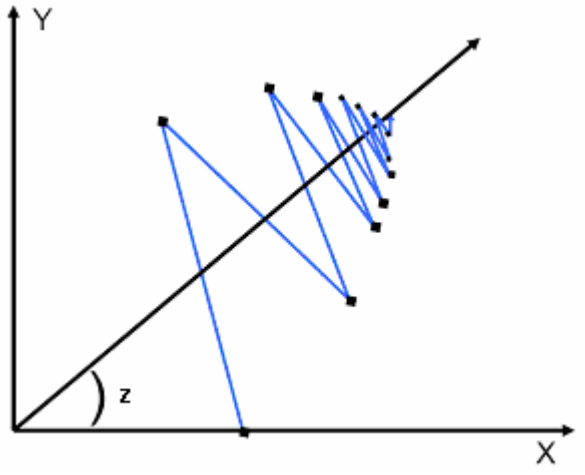
Rotasyonel modu; polar koordinat formundan kartezyen koordinat formuna dönüşüm için kullanılmaktadır. Rotasyonel modunda sinüs ve kosinüs fonksiyonu hesaplamaları yapılmaktadır. Ayrıca rotasyonel modunda z residue değeri sıfır değerine yakınsayana kadar işlemler sürdürülür. Şekil 2-4'de rotasyonel modu, Şekil 2-5'de ise rotasyonel modunun iteratif yakınsama eğrisi gösterilmiştir.

Rotasyonel modu girişleri: Başlangıç vektörü koordinat bileşenleri, rotasyon açısı

Rotasyonel modu çıkışlar: Yeni vektörün koordinat bileşenleri



Şekil 2 - 4. Rotasyonel modu vektör rotasyonu



Şekil 2 - 5. Rotasyonel modu iteratif yakınsama eğrisi

Birim çemberden yararlanılarak rotasyonel modundaki işlem adımları aşağıdaki gibi açıklanabilir.

Rotasyonel Mod İşlem Adımları:

$x_0 = 1,\ y_0 = 0$ ve $z = \theta$ başlangıç değerleri için:

- (1, 0) başlangıç vektörünü z kadar döndür. İteratif işlem adımları boyunca z değerini sıfıra değerine yakınsat. ($z_n \cong 0$)
- Döndürülmüş vektörün yeni x ve y koordinat bileşenlerini hesapla.
- $\cos(z) = x$ ve $\sin(z) = y$ atamalarını yap.

n iterasyon sonrasında x, y ve z değerleri Denklem (2.20)'deki gibi ifade edilebilir.

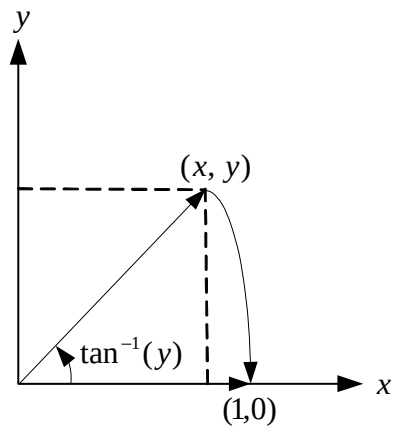
$$\begin{aligned} x_n &= A_n.[x_0.\cos(z_0) - y_0.\sin(z_0)] = \cos(z_0) \\ y_n &= A_n.[y_0.\cos(z_0) + x_0.\sin(z_0)] = \sin(z_0) \\ z_n &= 0 \end{aligned} \tag{2.20}$$

2.3.2. Vektörel Modu

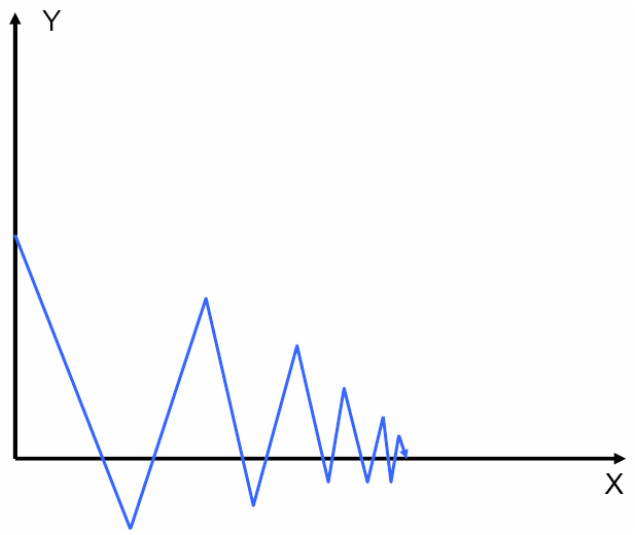
Vektörel modu; kartezyen koordinat formundan polar koordinat formuna dönüşüm için kullanılmaktadır. Vektörel modunda verilen kartezyen koordinat bileşenlerinden vektörün genlik ve açı bilgisi hesaplanmaktadır. Vektörel modunda y değeri sıfır değerine yakınsayana kadar işlemler sürdürülür. Şekil 2-6'da vektörel modu, Şekil 2-7'de ise vektörel modunun iteratif yakınsama eğrisi gösterilmiştir.

Vektörel modu girişleri: Başlangıç vektörü koordinat bileşenleri

Vektörel modu çıkışlar: Vektör uzunluğu ve açı



Şekil 2 - 6. Vektörel modu vektör rotasyonu



Şekil 2 - 7. Vektörel modu iteratif yakınsama eğrisi

Birim çemberden yararlanılarak vektörel modundaki işlem adımları aşağıdaki gibi açıklanabilir.

Vektörel Mod İşlem Adımları:

$x_0 = x$, $y_0 = y$ başlangıç değerleri için:

- (x, y) başlangıç vektör koordinatları için başlangıç vektörünü y=0 olana kadar döndür. İteratif işlem adımları boyunca y değerini sıfıra yakınsat. ($y_n \cong 0$)
- Vektörün dönüş açısını hesapla.
- Dönüş açısı = $\arctan(y)$ atamasını yap.

n iterasyon sonrasında x, y ve z değerleri Denklem (2.21)'deki gibi ifade edilebilir.

$$\begin{aligned} x_n &= A_n \cdot \sqrt{x_0^{\,2} + y_0^{\,2}} \\ y_n &= 0 \\ z_n &= z_0 + \tan^{-1}\left(y_0 / x_0\right) \end{aligned} \qquad (2.21)$$

Bu tez çalışmasında rotasyonel CORDIC modu üzerinde yoğunlaşılmıştır.

2.4. Rotasyon Tiplerine Göre CORDIC Denklemleri

Farklı rotasyon tiplerinin birleştirilmesi ve uygun değer atamalarının yapılmasıyla, çok sayıda birleşik fonksiyon hesaplanabilir. Ayrıca hesaplanan değerlerin uygun başlangıç değerleri ile çarpma, toplama, ters alma gibi işlemlerle kombine edilmesi sonucu Tablo 2-1'de tanımlanmış olan önemli fonksiyonlar geliştirilebilir.

Fonksiyon	**Eşitlik**	**Modlar**
tan(z)	$\sin(z) / \cos(z)$	$m = 1, 0$
tanh(z)	$\sinh(z) / \cosh(z)$,	$m = -1 , 0$
exp(z)	$\sinh(z) + \cosh(z)$	$m = -1,\ x = y = 1$
$\log_e(W)$	$2.\tanh^{-1}(Y/X)$	$m = -1, X = W + 1, Y = W - 1$
$\sqrt{W}$	$\sqrt{X^2 - Y^2}$	$m = 1,\ X = W + 1/4, Y = W - 1/4$

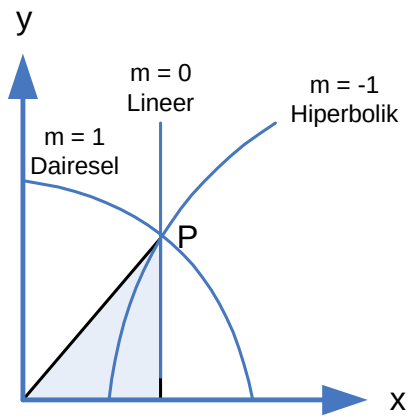
Tablo 2 - 1. Birleşik fonksiyonlar ve mod seçimleri

Tablo 2-2'de rotasyon tipi seçim parametresinin (m) almış olduğu değerlere karşılık gelen rotasyon fonksiyonları tanımlanmıştır.

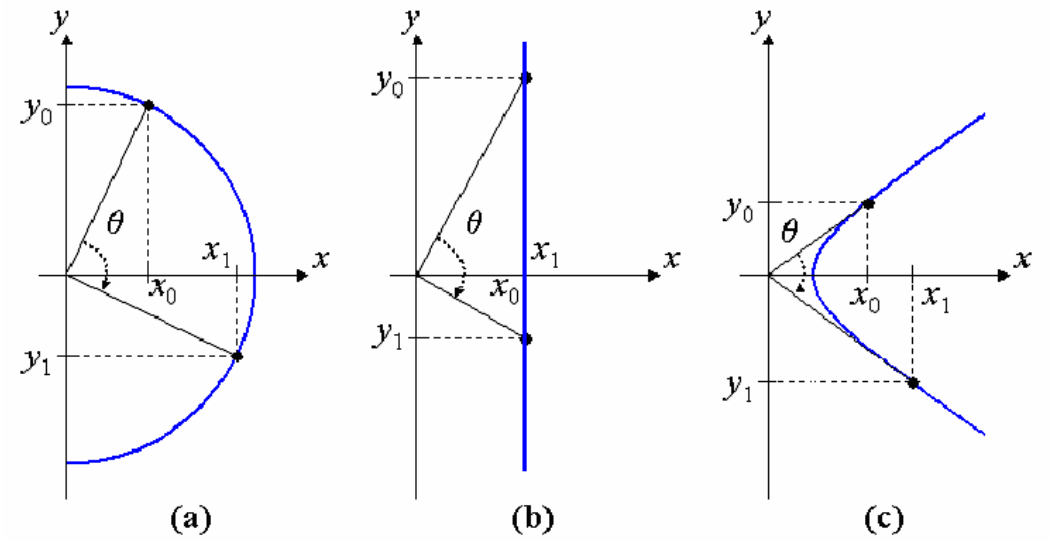
Rotasyon Tipi	**m**	**Rotasyon fonksiyonu** (i: iterasyon indeksi)
Dairesel Rotasyonlar	1	$\tan^{-1}(2^{-i})$
Lineer Rotasyonlar	0	2^{-i}
Hiperbolik Rotasyonlar	-1	$\tanh^{-1}(2^{-i})$

Tablo 2 - 2. Farklı rotasyon tipleri

Şekil 2-8'de farklı rotasyon tiplerinin kartezyen koordinat sisteminin ilk quadrantı boyunca karakteristik eğrileri gösterilmiştir. Şekil 2-9'da ise farklı rotasyon tiplerinin kartezyen koordinat sisteminin birinci ve dördüncü quadrantları boyunca karakteristik eğrileri vektör koordinat bileşenleri ve hedef açısı göz önüne alınarak detaylı bir şekilde gösterilmiştir.



Şekil 2 - 8. Farklı rotasyon modlarına ait karakteristik eğrileri



Şekil 2 - 9. Farklı rotasyon modlarına ait karakteristik eğrilerin detaylı gösterimi
(a) Dairesel rotasyonlar, (b) Lineer rotasyonlar, (c) Hiperbolik rotasyonlar

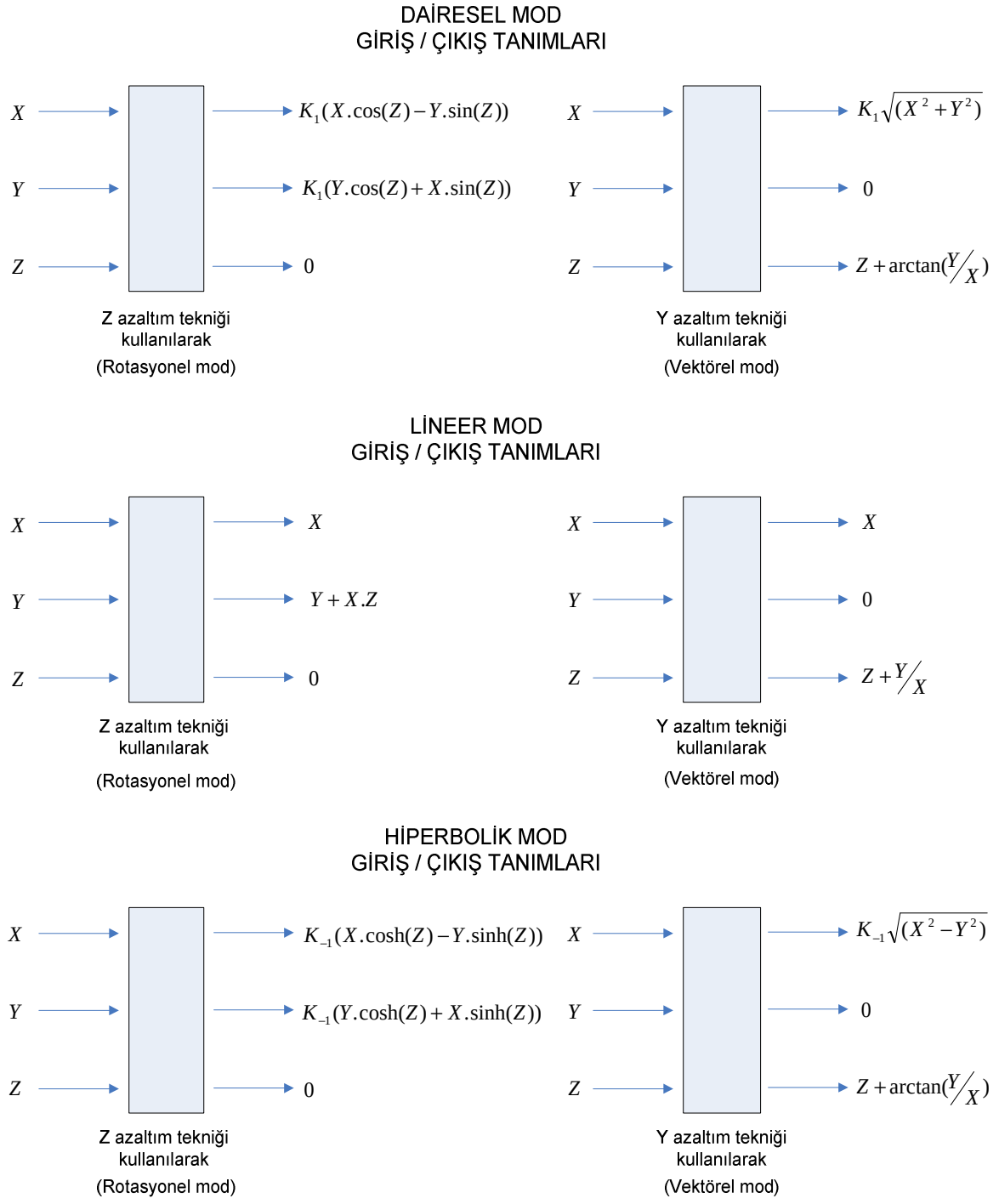
Rotasyon tiplerinin her biri rotasyonel ve vektörel modlarda çalışabilmektedir. Her iki mod için de rotasyonlar aynı iterasyon denklemlerini kullanmaktadır. Yani dairesel rotasyon tipi için rotasyonel modu ve vektörel modu iterasyon denklemleri aynıdır. Benzer durum diğer iki rotasyon tipi (lineer ve hiperbolik rotasyonlar) için de geçerlidir. Her üç rotasyonel tipi için rotasyonel ve vektörel moda ait iterasyon denklemleri Tablo 2-3'de verilmiştir.

$$x_{i+1} = x_i - m.y_i.r_i.2^{-i}$$
$$y_{i+1} = y_i + x_i.r_i.2^{-i}$$
$$z_{i+1} = z_i - r_i.f_i$$

m = 1 **Dairesel** **Rotasyonlar**	**m = 0** **Lineer Rotasyonlar**	**m = -1** **Hiperbolik Rotasyonlar**
$x_{i+1} = x_i - y_i.r_i.2^{-i}$ $y_{i+1} = y_i + x_i.r_i.2^{-i}$ $z_{i+1} = z_i - r_i.\tan^{-1}(2^{-i})$	$x_{i+1} = x_i$ $y_{i+1} = y_i + x_i.r_i.2^{-i}$ $z_{i+1} = z_i + r_i.(2^{-i})$	$x_{i+1} = x_i + y_i.r_i.2^{-i}$ $y_{i+1} = y_i + x_i.r_i.2^{-i}$ $z_{i+1} = z_i - r_i.\tanh^{-1}(2^{-i})$
Rotasyonel Modu	**Rotasyonel Modu**	**Rotasyonel Modu**
$x_n = A_n.[x_0.\cos(z_0) - y_0.\sin(z_0)]$ $y_n = A_n.[y_0.\cos(z_0) + x_0.\sin(z_0)]$ $z_n = 0$ $A_n = \prod_{i=0}^{n}\sqrt{1+2^{-2i}}$ $r_i = \begin{cases} -1 & , \quad z_i < 0 \\ +1 & , \quad diger\ durumlar \end{cases}$	$x_n = x_0$ $y_n = y_0 + x_0.z_0$ $z_n = 0$ $r_i = \begin{cases} -1 & , \quad z_i < 0 \\ +1 & , \quad diger\ durumlar \end{cases}$	$x_n = A_n.[x_0.\cosh(z_0) + y_0.\sinh(z_0)]$ $y_n = A_n.[y_0.\cosh(z_0) + x_0.\sinh(z_0)]$ $z_n = 0$ $A_n = \prod_{n}\sqrt{1+2^{-2i}} \cong 0.80$ $r_i = \begin{cases} -1 & , \quad z_i < 0 \\ +1 & , \quad diger\ durumlar \end{cases}$
Vektörel Modu	**Vektörel Modu**	**Vektörel Modu**
$x_n = A_n.\sqrt{x_0^{\,2} + y_0^{\,2}}$ $y_n = 0$ $z_n = z_0 + \tan^{-1}(y_0 / x_0)$ $A_n = \prod_{i=0}^{n}\sqrt{1+2^{-2i}}$ $r_i = \begin{cases} +1 & , \quad y_i < 0 \\ -1 & , \quad diger\ durumlar \end{cases}$	$x_n = x_0$ $y_n = 0$ $z_n = z_0 - y_0 / x_0$ $r_i = \begin{cases} +1 & , \quad y_i < 0 \\ -1 & , \quad diger\ durumlar \end{cases}$	$x_n = A_n.\sqrt{x_0^{\,2} - y_0^{\,2}}$ $y_n = 0$ $z_n = z_0 + \tanh^{-1}(y_0 / x_0)$ $A_n = \prod_{n}\sqrt{1+2^{-2i}}$ $r_i = \begin{cases} +1 & , \quad y_i < 0 \\ -1 & , \quad diger\ durumlar \end{cases}$
İterasyon denklemleri her iki mod için de aynıdır.	İterasyon denklemleri her iki mod için de aynıdır.	İterasyon denklemleri her iki mod için de aynıdır.

Tablo 2 - 3. Farklı rotasyon tiplerine ait iterasyon denklemleri

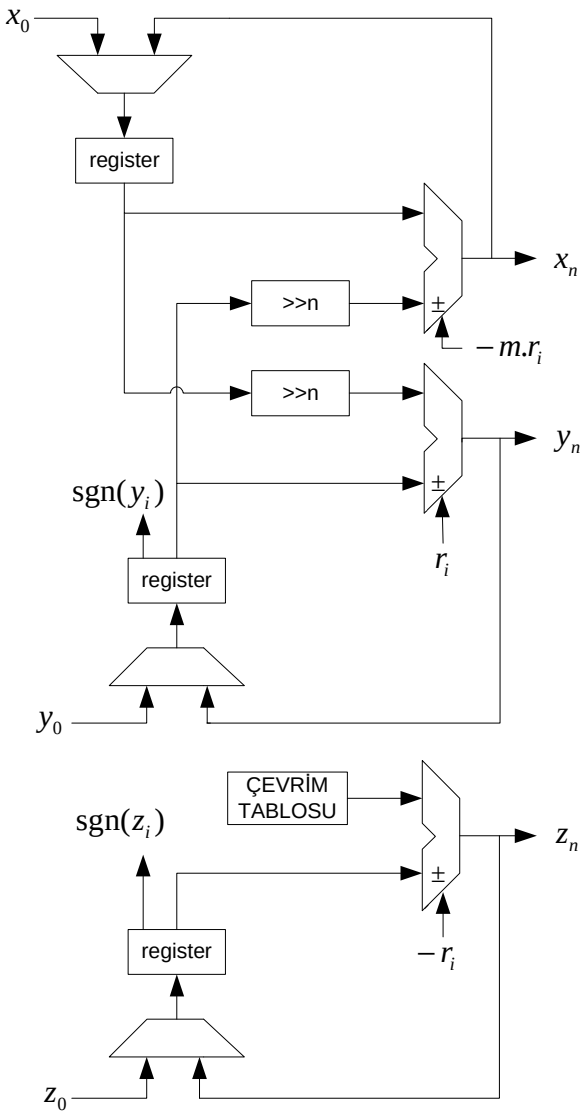
Dairesel, lineer ve hiperbolik rotasyonel modlar için giriş-çıkış tanımları rotasyonel ve vektörel modları ayrı ayrı ifade edilerek Şekil 2-10'da verilmiştir.



Şekil 2 - 10. Farklı rotasyon modları için giriş/çıkış tanımları

2.5. İteratif CORDIC Yapısı

Bir bit, bir iterasyona karşılık gelmektedir. Denklem (2.19)'un tüm rotasyon tipleri için en genel haline karşılık gelen iteratif CORDIC yapısı Şekil 2-11'deki gibi olacaktır.



Şekil 2 - 11. İteratif CORDIC yapısı

Büyük değerdeki θ hedef açısını hesaplayabilmek için α_i mikro rotasyon açısı çok küçük seçilerek işlemin doğruluk yüzdesi artırılabilir. Ancak bu durumda iteratif çalışan CORDIC iterasyon sayısının da artacağı göz önüne alınmalıdır. Tasarımın doğruluk derecesi artırılmaya çalışılırken donanım sınırları ve sistem performansı arasında hassas bir denge kurulmalıdır.

2.6. Pipeline CORDIC Yapısı

2.6.1. Pipeline Yapı Kullanılanılarak Tasarlanan Sistemin Avantajları

Bir mikrodenetleyici veya DSP gerçek anlamda paralel işlem yeteneğine sahip değilken veya sınırlı seviyede pipeline çalışabilirken FPGA içerisinde gerçek paralel işlem birimlerinin oluşturulması mümkündür. Birbirinden bağımsız ve eşzamanlı çalışan işlem birimlerinin bir yonga üzerinde gerçeklenmesi performans açısından tasarımcıya avantaj sunmaktadır. Bu durum [14]'de ayrıntılı olarak ele alınmıştır. Pipeline çalışan mimarilerde daha az saat çevriminde daha çok iş yapılarak operasyonlar hızlandırılır ve işlem hacmi artırılmış olur. Pipeline yapı işlemi bir otomobil montaj hattında olduğu gibi işlemleri küçük alt birimlere ayırarak çalışma sürecindeki tıkanmaları önler.

2.6.2. Pipeline Yapı Kullanılanılarak Tasarlanan Sistemin Dezavantajları

- Pipeline yapıya sahip bir sistemin yönerge gecikmesi(latency), pipeline olmayan yapıya göre daha fazladır. Bu da sisteme ekstra flip-floplar eklenmesinden kaynaklanır.

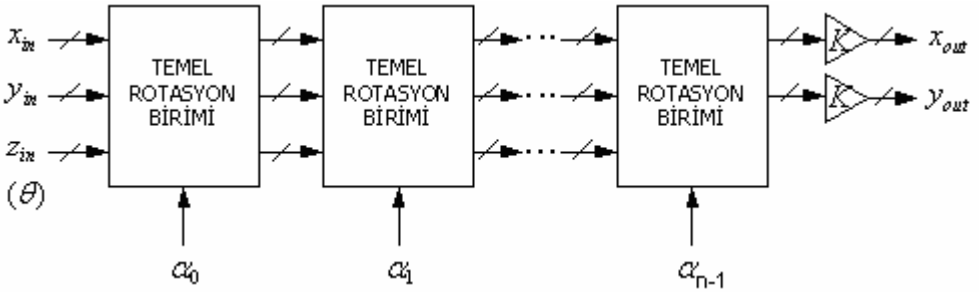
- Pipeline mimarilerde ALU tasarımı daha karmaşıktır.

- Pipeline yapılarda bir işlemi alt bölümlere ayırmak diğer yapılara göre daha zordur.

- Donanımda kullanılan alan pipeline çalışmayan mimarilere göre daha fazladır. Bu ise güç tüketimini beraberinde getirmektedir. Alt rotasyonlar için benzer donanımın tekrarlanması maliyeti artıracaktır.

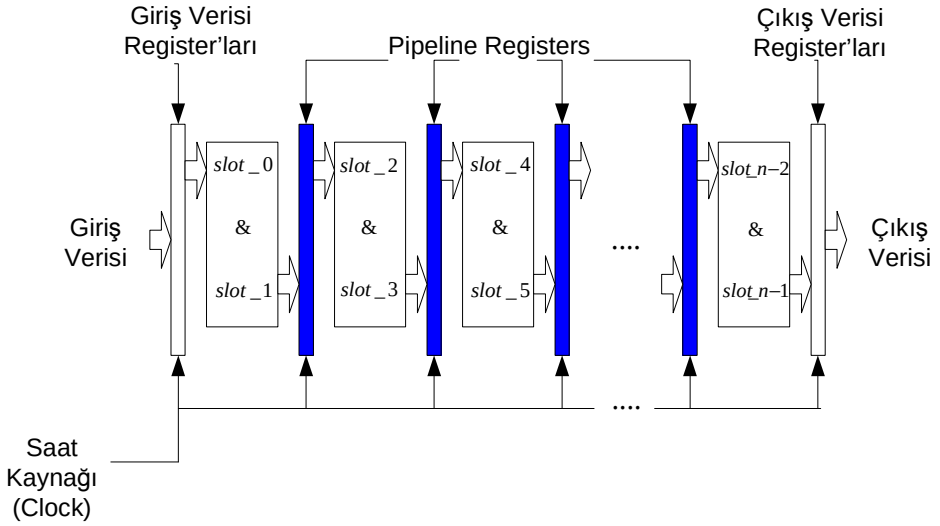
2.6.3. Pipeline Yapı İşlem Aşamaları

Tüm iterasyonlar paralel olarak pipeline yapısında gerçekleştirilir. 'n' sayıda slot için pipeline yapı formunda yürütülen dairesel CORDIC algoritması genel hatları ile Şekil 2-12'de gösterilmiş olup, pipeline mimari Şekil 2-13'de daha ayrıntılı olarak ifade edilmiştir.



Şekil 2 - 12. Pipeline yapı formunun n sayıda slot için gösterimi

Pipeline yapı sayesinde, her saat çevriminde (clock cycle) bir adet CORDIC dönüşüm operasyonu yürütülebilmektedir.



Şekil 2 - 13. n sayıda slot için pipeline mimari yapısı

3. TRİGONOMETRİK FONKSİYONLARIN HESAPLANMASI

Sin, cos, tan vb.. trigonometrik fonksiyonlar çok sayıda problemin çözümünde kullanılmaktadır.

- **Sinyal İşleyiciler:** Dönüşümler, filtreleme, radar uygulamaları (radar plan mevki gösterici alanı (PPI skope) arayüz yazılımları, radar video plot çıkartma sistemleri vb..)
- **Robotik Uygulamaları:** Hareket kestirimi, ortam geometrisinin hesaplanması, synchro resolver uygulamaları (radar anteni yan, yükseliş, mesafe yönlenme hesaplamaları)
- **Lineer Sistemler:** Kontrol uygulamaları

Yukarıda ifade edilen uygulama alanlarından birkaçı altıncı bölümde ele alınmış olup, CORDIC algoritmalarının bu uygulamalarda nasıl kullanıldığı ve hangi avantajları beraberinde getirdiği açıklanmıştır.

Trigonometrik fonksiyonlar farklı hesaplama teknikleri kullanılarak üretilebilirler. Bu bölümde sinüs ve kosinüs trigonometrik fonksiyonlarının hesaplanmasında aşağıda listelenen üç teknik açıklanacaktır.

Bu teknikler:

1. Çevrim tablosu (Look-up table/ROM) kullanılarak
2. Polinom yakınsaması (Taylor serisi) kullanılarak
3. CORDIC algoritması kullanılarak

3.1. Çevrim Tablosu Kullanılarak Trigonometrik Fonksiyonların Hesaplanması

Çok sayıda temel fonksiyon sıklıkla çevrim tablosu kullanılarak hesaplanabilmektedir. Bunlardan biri olan trigonometrik fonksiyon hesaplamalarını içeren uygulamalarda da çevrim tabloları kullanılmaktadır.

- ROM çevrim tablosu uygulamaları aynı sonucu veren donanımsal uygulamalar ile harcanan işlem zamanı bakımından karşılaştırıldığında, ROM uygulamalarının en hızlı hesaplama yöntemlerinden biri olduğu söylenebilir.
- ROM kullanımı donanımda hafıza kullanımı ihtiyacını doğurur.
- Sonucun doğruluğu sınırlandırılmıştır.
- Sonucun doğruluğu ROM boyutu ile doğrudan ilişkilidir. Yüksek doğrulukta sonuçlara çevrim tablosu kullanılarak ulaşmak mümkünken, doğruluk oranı arttıkça donanımda kullanılan uçucu olmayan bellek alanının (nonvolatile memory) da arttığı gerçeği göz ardı edilmemelidir. Tablo boyutunda hafıza tasarrufuna gidilmesi durumunda, sonucun doğruluğu düşecektir.
- Çevrim tablosunda tutulan açı değerleri radyan cinsindedir.

3.2. Polinom Yakınsaması İle Trigonometrik Fonksiyonların Hesaplanması

Taylor serileri

- Kayan nokta (floating point) işlem gerektirir.
- Iteratif çalışma yapısına sahiptir
- Yavaş çalışırlar
- Hesap hızının önemsiz olduğu uygulamalarda kullanılırlar.

Sinüs ve kosinüs fonksiyonları için Taylor serisi yaklaşımı

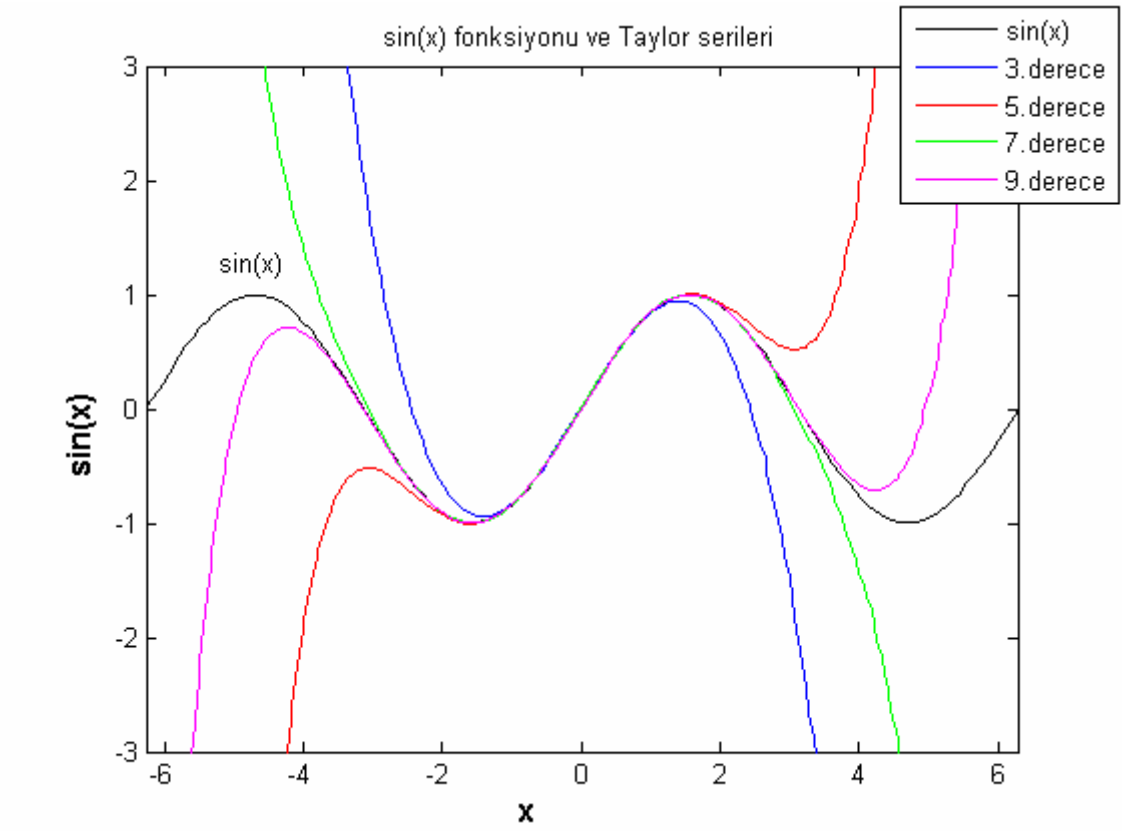
Taylor serileri polinom yaklaşımını kullanmakta olup, karmaşık birçok fonksiyonun polinom yakınsaması ile elde edilmesinde kullanılır. Trigonometrik fonksiyonların hesaplanmasında Taylor serisi yaklaşımı oldukça popülerdir.

MATLAB'de sinüs fonksiyonu için $(-2\pi, 2\pi)$ aralığında x değişkeninin 3. ,5. , 7. ve 9. kuvvetlerinde polinom yakınsaması kullanılarak üretilen fonksiyonlar ve sinüs(x) fonksiyonu simüle edilmiş olup, MATLAB çıktısı Şekil 3-1'de gösterilmiştir. Kosinüs fonksiyonu için de MATLAB'de $(-2\pi, 2\pi)$ aralığında x değişkeninin 2., 4. ,6. ve 8. kuvvetlerinde polinom yakınsaması kullanılarak üretilen fonksiyonlar ve kosinüs(x) fonksiyonu simüle edilmiştir. MATLAB çıktısı Şekil 3-2'de gösterilmiştir.

MATLAB'de simülasyon sırasında Taylor serisi açılımının n. kuvvetler için etksininin kolaylıkla görülebilmesi için x ekseni $(-2\pi, 2\pi)$ aralığında düzenlenmiştir.

Sinüs fonksiyonu için Taylor serisi Denklem (3.1)'deki gibi olacaktır.

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \text{ tüm x değerleri için} \qquad (3.1)$$

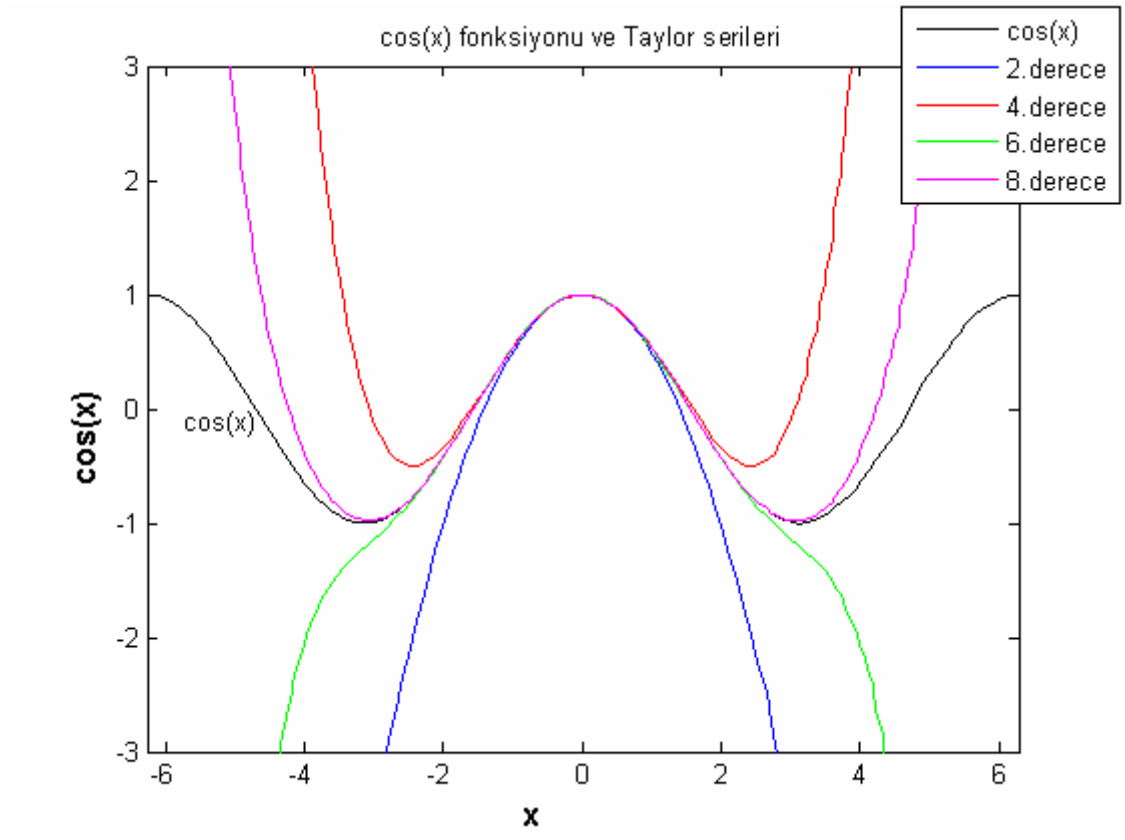


Şekil 3 - 1. Sinus fonskiyonu ve Taylor serileri

Şekil 3-1'de görüldüğü üzere, Taylor serisi mertebesi arttıkça sinyaller sin(x) sinyaline doğru yakınsamaktadır.

Kosinüs fonksiyonu için Taylor serisi Denklem (3.2)'deki gibi olacaktır.

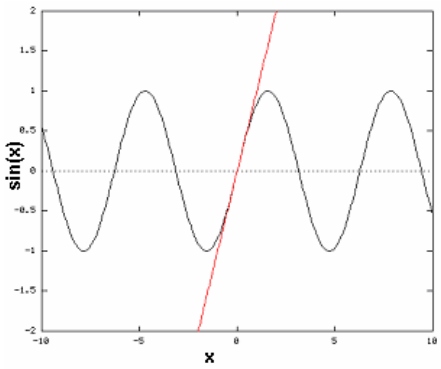
$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \quad \text{tüm x değerleri için} \qquad (3.2)$$



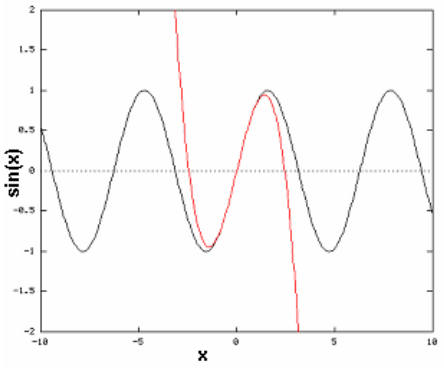
Şekil 3 - 2. Kosinus fonskiyonu ve Taylor serileri

Şekil 3-2'de görüldüğü üzere, Taylor serisi mertebesi arttıkça sinyaller cos(x) sinyaline doğru yakınsamaktadır.

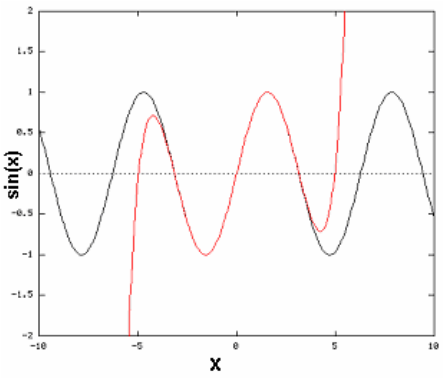
Sinüs fonksiyonu için Taylor serisi yaklaşımını 15. dereceye kadar MATLAB'de simüle edecek olursak sonuçların sinüs fonksiyonuna göre değişimi Şekil 3-3'de gösterilmiştir.



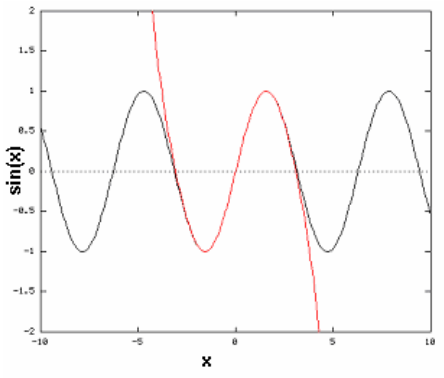
Y = X



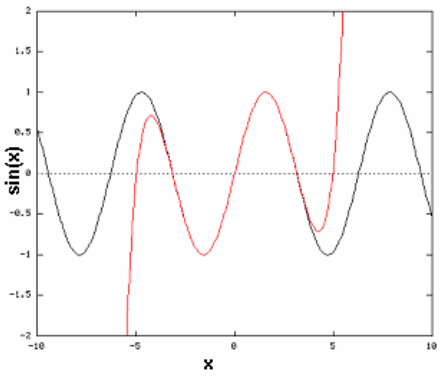
Y = X - X^3/ 3!



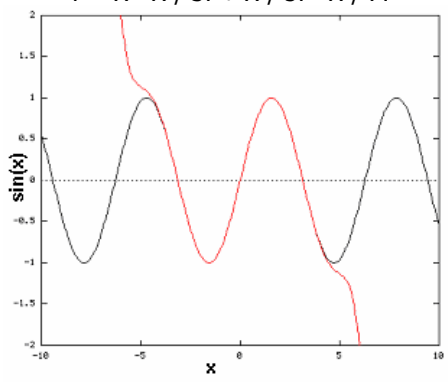
Y = X - X^3/ 3! + X^5/ 5!



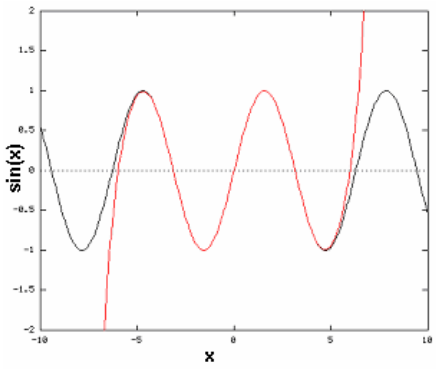
Y = X - X^3/ 3! + X^5/ 5! - X^7/ 7!



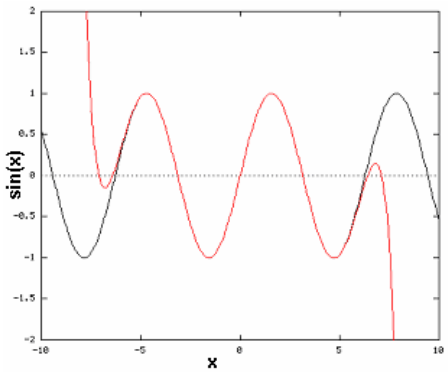
Y = X - X^3/ 3! + X^5/ 5! - X^7/ 7! + X^9/ 9!



Y = X - X^3/ 3! + X^5/ 5! - X^7/ 7! + X^9/ 9! - X^{11}/ 11!



Y = X - X^3/ 3! + X^5/ 5! - X^7/ 7! + X^9/ 9! - X^{11}/ 11! + X^{13}/ 13!

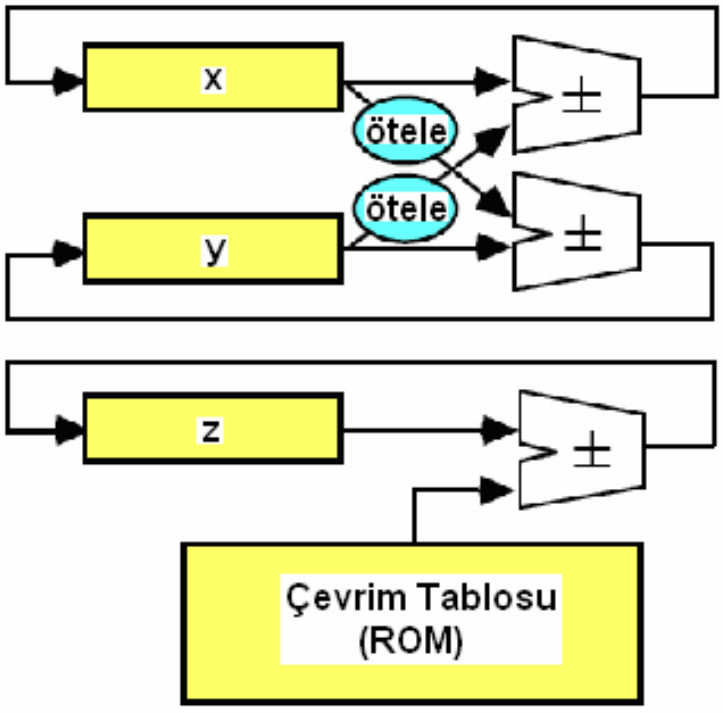


Y = X - X^3/ 3! + X^5/ 5! - X^7/ 7! + X^9/ 9! - X^{11}/ 11! + X^{13}/ 13! - X^{15}/ 15!

Şekil 3 - 3. Sinüs fonksiyonu için Taylor serileri

Taylor serisi açılımının n derecede temsil edildiğini varsayacak olursak, n arttıkça fonksiyonun doğruluğu da giderek artacaktır. Sinüs fonksiyonu için Taylor serisi açılımının n derecesi arttıkça fonksiyonun yüksek doğruluğa doğru yakınsaması Şekil 3-3'de gösterilmiştir. Taylor serisi açılımında n=1'den 15'e doğru Taylor kuvvetleri arttıkça şekilden de görüldüğü üzere yakınsayan sinyal, gerçekte olması gereken sinüs(x) sinyaline doğru yaklaşmaktadır. Hatasız bir polinom yakınsaması hiçbir zaman gerçeklenemeyeceği için Taylor serisi sınırlandırılmalıdır. Beşinci bölümde açıklanan "GUICORDIC" MATLAB programında kullanıcı n mertebesini kendisi seçerek bu sınırlandırmayı yapmakta ve sinüs, kosinüs değerlerini elde edebilmektedir.

3.3. CORDIC Algoritması Kullanılarak Trigonometrik Fonksiyonların Hesaplanması



Şekil 3 - 4. Trigonometrik fonksiyonların hesaplanmasında çevrim tablosu kullanımı

CORDIC algoritmasında her bir iterasyon için 2'nin negatif kuvvetlerinin arktanjant değerleri olan $\alpha_i = \tan^{-1}(2^{-i})$ mikro rotasyon açı değerleri önceden hesaplanarak küçük bir çevrim tablosunda tutulmaktadır.

Sinüs ve kosinüs fonksiyonlarının çözümlenmesinde kullanılan iteratif algoritma yürütülürken her bir iterasyonda çevrim tablosunda ilgili iterasyona ait adresde bulunan mikro rotasyon açı değeri hesaba katılmaktadır. Her bir iterasyon için 2'nin negatif kuvvetlerinin arktanjant değerleri aşağıda i=(0:15) aralığı için ifade edilmiş olup ayrıca K ölçek sabitinin her bir iterasyonda alması gereken değerlerde i=(0:15) aralığı için radyan cinsinden ifade edilmiştir. Tablo 3-1'de ise n=10 iterasyon adımı için çevrim tablosu açı değerleri derece cinsinden verilmiştir.

açılar = $\arctan(2^{-i})$ = [...
0.78539816339745 0.46364760900081 0.24497866312686 0.12435499454676 ...
0.06241880999596 0.03123983343027 0.01562372862048 0.00781234106010 ...
0.00390623013197 0.00195312251648 0.00097656218956 0.00048828121119 ...
0.00024414062015 0.00012207031189 0.00006103515617 0.00003051757812];
K ölçek değerleri = [...
0.70710678118655 0.63245553203368 0.61357199107790 0.60883391251775 ...
0.60764825625617 0.60735177014130 0.60727764409353 0.60725911229889 ...
0.60725447933256 0.60725332108988 0.60725303152913 0.60725295913894 ...
0.60725294104140 0.60725293651701 0.60725293538591 0.60725293510314];

Adresler (i)	**Açı** $\left(\alpha_i = \tan^{-1}(2^{-i})\right)$ **(Derece)**	$\tan(\alpha_i) = 2^{-i}$	2^{-i}
0	45	1	2^{-0}
1	26.6	0.5	2^{-1}
2	14.0	0.25	2^{-2}
3	7.1	0.125	2^{-3}
4	3.6	0.0625	2^{-4}
5	1.8	0.03125	2^{-5}
6	0.9	0.015625	2^{-6}
7	0.4	0.0078125	2^{-7}
8	0.2	0.00390625	2^{-8}
9	0.1	0.001953125	2^{-9}

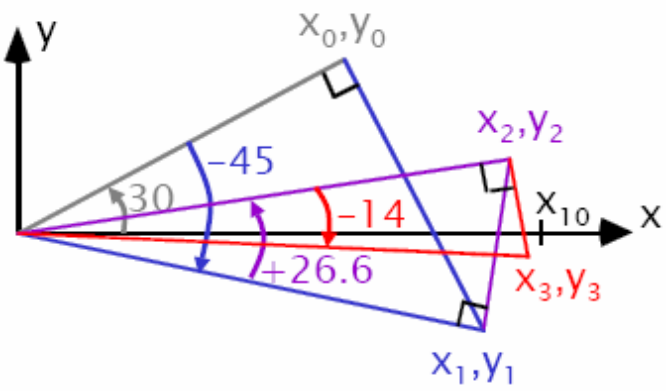
Tablo 3 - 1. Çevrim tablosu değerlerinin hesaplanması

Herhangi bir θ açısı n=10 iterasyon adımı için; $\theta = \pm\alpha_0 \pm \alpha_1 \pm \alpha_2 \pm \cdots \pm \alpha_9$ şeklinde hesaplanır.

Algoritma:

- **Mod:** Rotasyonel modu: "Her iterasyonda z_{i+1} açısı $z_{i+1} = z_i - \mathrm{r_i}.\alpha_i$ eşitliğine uygun olarak sıfır (0) değerine doğru yakınsatılmaya çalışılır."

- **Başlangıç yapılandırması:** $x = 0.607253$, $y = 0$, $z = \theta$ **(hedef açısı)**
- $For\ i = 0 \rightarrow n$
  - $r_i = \begin{cases} 1, & z > 0\ iken \\ -1, & z \le 0\ iken \end{cases}$
  - $x_{i+1} = x_i - r_i.2^{-i}.y_i$
  - $y_{i+1} = y_i + r_i.2^{-i}.x_i$
  - $z_{i+1} = z_i - r_i.\alpha_i$
- **Sonuç:** $x_n = \cos(\theta)$, $y_n = \sin(\theta)$
- **Doğruluk:** $n\ bit\ \left(\tan^{-1}(2^{-i}) \approx 2^{-i}\right)$



Şekil 3 - 5. Klasik CORDIC vektör rotasyonları

ÖRNEK-1:

Giriş verileri: $(x, y)^T$ ve rotasyon açısı $\theta = 30^O$ açısı

Yukarıdaki giriş verilerine göre Klasik CORDIC algoritması rotasyon adımlarını maksimum 10 iterasyon adımı için yürütecek olursak;

i	$z_{i+1} = z_i - \mathrm{r_i}.\alpha_i$
0	$\alpha_0 = +30 - 45.0^0 = -15.0$
1	$\alpha_1 = -15.0^0 + 26.6^0 = +11.6^0$
2	$\alpha_2 = +11.6^0 - 14.0^0 = -2.4^0$
3	$\alpha_3 = -2.4^0 + 7.1^0 = +4.7^0$
4	$\alpha_4 = +4.7^0 - 3.6^0 = +1.1^0$
5	$\alpha_5 = 1.1^0 - 1.8^0 = -0.7^0$
6	$\alpha_6 = -0.7^0 + 0.9^0 = +0.2^0$
7	$\alpha_7 = +0.2^0 - 0.4^0 = -0.2^0$
8	$\alpha_8 = -0.2^0 + 0.2^0 = 0.0^0$
9	$\alpha_9 = 0.0^0 - 0.1^0 = -0.1^0$

Tablo 3 - 2. Klasik CORDIC rotasyon adımları

Çıkış verileri:

$\left(x^{'}, y^{'}\right)^T$, $x' = x - y.\tan(30^0)$, $y' = y + x.\tan(30^0)$

$z_{i+1} = z_i - \mathrm{r_i}.\alpha_i$ için: $r_i = [1, -1, 1, -1, 1, 1, -1, 1, -1, 1]$, $i \in \{0, 1, 2, \cdots, 9\}$

4. FARKLI CORDIC MİMARİLERİNİN İNCELENMESİ

4.1. Çevrim Tablosu Kullanan CORDIC Mimarileri

Klasik CORDIC algoritması [1] küçük bir çevrim tablosu kullanımını gerektirmektedir.

Trigonometrik fonksiyon hesaplamalarının doğruluğu algoritmanın iterasyon sayısı artırılarak iyileştirilebilir. Giderek artan iterasyon sayıları ile hesaplamanın doğruluk derecesi artarken, çevrim tablolarının (ROM) kullanılması, donanımda kullanılan alanı arttırdığından algoritmanın verimliliği önemli ölçüde sınırlanır. Bu nedenle "donanım boyutu" ve "doğruluk" arasındaki hassasiyete çok dikkat edilmelidir.

Bu iki karakteristik parametre sistemin verimliliğini ciddi bir şekilde etkiler. Donanım boyutu büyüdükçe, sistemin maliyeti de giderek artacaktır. O halde iterasyon sayısı ve doğruluk göz önüne alınması gereken iki önemli parametredir. İteratif çalışan algoritma için bir bit, bir iterasyon adımına karşılık gelmektedir.

[16]'da ROM kullanan CORDIC algoritmaları detaylı bir şekilde açıklanmıştır.

Tablo 4-1'de bit sayısına göre (iterasyon sayısı), ROM depolama gereksinim miktarları verilmiştir. Tablo 4-1'den görüleceği üzere bit-sayısı lineer arttıkça donanımda kullanılan alan eksponansiyel olarak artmaktadır. Donanımda kullanılan alanının eksponansiyel olarak artması tasarımcı için istenmeyen bir durumdur. Bu durum tasarımın uygulanabilirliğini zorlamakla beraber maliyeti de artıtacaktır.

Bit Sayısı	Tablo Girişleri	MBytes
8	256	0.0002
12	4096	0.0059
16	65536	0.125
20	1048576	2.5
24	16777216	48
28	268435456	896
32	4294967296	16385

Tablo 4 - 1. Çevrim tablosu (ROM) depolama gereksinimleri

4.1.1. Klasik CORDIC Algoritması İçin Donanımda Kullanılan Alan Gereksinimi

Klasik CORDIC uygulamasında [1] (16-bit için) açı yakınsaması için: 48 adet 16-bit toplayıcı (adder)

Ölçek faktörü kullanımı için gerekli öteleme ve toplama işlemleri için: 32 adet 16-bit toplayıcı (adder) gereklidir.

Toplam alan = 80 adet 16-bit toplayıcı (adder) = 1280 tam toplayıcı (full adder) gereklidir.

4.1.2. Dezavantajları

- Ölçek faktörünün donanımda büyük bir alan kullanımına neden olması,

hala istenilen yüksek işlem hızına ulaşılamamış olması,

- sınırlı yakınsama oranları,
- ölçek faktörünün bulunması ve iterasyon sayısına bağımlı oluşu,
- hesaplamaları gerçekleştirebilmek için operasyonun ilk basamağındaki başlangıç değerinin tamamı ile bilinmesi gerekliliği,
- bu amaçla logaritma çevrim tablosu kullanılma zorunluluğu.
- kayan noktalı sayılar, çevrim tablosunda kullanıldığı taktirde çevrim tablosu girişi ve depolama için ihtiyaç duyulan kapasite ciddi oranlarda artış gösterecektir.

Önemli bir karar noktası: Kayan nokta ve sabit nokta işlemler arasındaki tercih oldukça önemlidir. Kayan noktalı çözüm, sabit noktalı operasyonlara göre en yüksek doğruluğu verirken, kayan noktalı operasyonlar için maliyet ciddi oranda fazla olacaktır.

4.2. Çevrim Tablosu Kullanmayan CORDIC Mimarileri

Çevrim tablolarının donanımda kullanılması ürün maliyeti ve donanımda kullanılan alan yönünden ciddi dezavantajlar getirmektedir. Bu dezavantajları ortadan kaldırmak amacı ile çevrim tablosuna ihtiyaç duymayan çeşitli matematiksel yaklaşım ve optimizasyon yöntemlerini kullanan algoritmaların geliştirilmesi yoluna gidilmiştir. Bu yaklaşımlardan en etkin olarak kullanılanı CORDIC algoritmaları olup, bu bölümde çevrim tablosunu mimarisinde tamamiyle kullanmayan CORDIC algoritmaları üzerinde durulacaktır. Bu tez çalışmasında analizleri yapılan çevrim tablosu kullanmayan CORDIC algoritmaları Tablo 4-2'de verilmiştir.

1	"Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC Algoritması"[9]
2	"Modifiye Edilmiş Gerçekte Ölçekleme Bağımsız Adaptif CORDIC Dönüştürücü Algoritması"[10]
3	"Düşük Güç Tüketen CORDIC Çekirdeği Algoritması"[11]
4	"Yüksek Doğruluklu Azaltılmış İterasyonlu CORDIC İşlemci Algoritması"[12]

Tablo 4 - 2. Çevrim tablosu kullanmayan CORDIC algoritmaları

4.2.1. Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC Algoritması

İyileştirilmiş tüm CORDIC algoritmaları Klasik CORDIC algoritmasından [1] türetilmiştirler. Bu yaklaşımın amacı; (i) güç açısından verimli, (ii) ölçek faktörü kompanzasyon probleminden bağımsız ve (iii) tüm koordinat uzayında yakınsama oranına sahip bir CORDIC işlemcisi geliştirmektir. “Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC” algoritmasında [9], ölçek faktörü Klasik CORDIC algoritmasının [1] aksine, yürütülen iterasyon sayısından tamamıyla bağımsızdır. Algoritmada kullanılan rotasyon denklemi ve ölçek faktörünün iterasyon sayısından bağımsız oluşu aritmetik hesaplamaların sayısını ciddi oranlarda azaltmaktadır. Bu durum düşük güç tüketimi avantajını ortaya çıkarmaktadır.

Klasik CORDIC algoritmasının [1] aksine “Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC” algoritmasında [9], θ hedef açısı, vektör yalnız bir yönde döndürülerek elde edilir (saat yönü ya da saat yönüne ters yönde). Böylece Klasik CORDIC [1]’ deki vektörün iki yönlü dönüşü yaklaşımından kaçınılmış olur.

Bu mantıkla Denklem (2.15) yeniden yazılacak olursa:

$$\theta = \alpha_0 + ... + \alpha_{n-1} = \sum_{i=0}^{n-1} \alpha_i \quad (r_i = +1) \text{ için} \tag{4.1}$$

Bu demektir ki hedeflenen açı, temel mikro rotasyon açılarının basitçe toplamının yakınsamasıdır.

Sinüs ve kosinüs fonksiyonları için Taylor serisi açılımlarını inceleyecek olursak;

$$\sin(\alpha_i) = \alpha_i - \alpha_i^3/3! + \alpha_i^5/5! - \cdots\cdots \tag{4.2}$$

$$\cos(\alpha_i) = 1 - \alpha_i^2/2! + \alpha_i^4/4! - \cdots\cdots \tag{4.3}$$

Çok küçük mikro rotasyon açıları için Taylor serisi açılımında α_i ifadesi 2^{-i} ifadesine yakınsanabilir. Bu koşul göz önüne alınarak Denklem (4.2) ve (4.3) yeniden düzenlenecek olursa:

$$\sin(\alpha_i) = 2^{-i} - 2^{-3i}/3! + 2^{-5i}/5! - \cdots\cdots \tag{4.4}$$

$$\cos(\alpha_i) = 1 - 2^{-2i-1} + 2^{-4i}/4! - \cdots\cdots \quad (4.5)$$

elde edilir.

Sinüs fonksiyonunun Taylor serisi açılımında, çok küçük açı değerleri için üçüncü derece ve üstündeki terimler ihmal edilebilir. Kosinüs fonksiyonunun Taylor serisi açılımında ise dördüncü derece ve üstündeki terimler ihmal edilebilir. Bu durumlar göz önüne alındığında;

$$\sin(\alpha_i) = 2^{-i} \quad (4.6)$$

$$\cos(\alpha_i) = 1 - 2^{-(2i+1)} \quad (4.7)$$

$$z_{i+1} = z_i - r_i.2^{-i} \quad (4.8)$$

elde edilir.

Denklem (4.6) ve (4.7), Denklem (2.11)'de tek bir rotasyon yönü için ($r_i = -1$) yerine konulursa, bir iterasyon adımı için "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" [9] rotasyonel denklemi elde edilir.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1-2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1-2^{-(2i+1)} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (4.9)$$

$$z_{i+1} = z_i - r_i.2^{-i}$$

b bit sayısı işlemcinin üzerinde işlem yapabildiği en uzun verinin bit sayısı (kelime uzunluğu) olmak üzere;

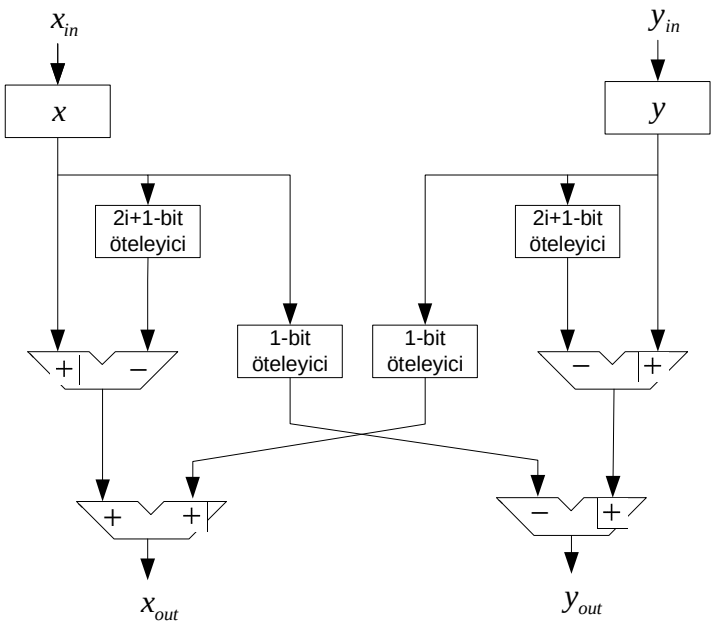
b sayıda iterasyon için Denklem (4.9) yeniden düzenlenecek olursak "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" [9] rotasyonel denkleminin en genel hali elde edilir.

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \prod_{i=0}^{b-1} \begin{bmatrix} 1-2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1-2^{-(2i+1)} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (4.10)$$

$$z_{i+1} = z_i - r_i.2^{-i}$$

Bu algoritmada θ hedef açısı; tek yönlü vektör rotasyonları uygulanarak, çevrim tablosu kullanılmaksızın elde edilmektedir. Vektör rotasyonları boyunca r_i '+1' değerine sahiptir.

Denklem (4.10), [9]'da tartışılan CORDIC algoritmasının çalışma biçimini ifade etmektedir. (4.10) nolu denklemde ifade edilen operasyonların yürütülebilmesi için gerekli veri yolu ve bileşenler Şekil 4-1'de gösterilmiştir.



Şekil 4 - 1. Temel rotasyon adımları

Önemli bir karşılaştırma:

(2.19) ve (4.10) notu denklemler bir iterasyon adımı için karşılaştırıldığında Denklem (2.19) notu denklemde $\alpha_i = \tan^{-1}(2^{-i})$ mikro rotasyon açı değerleri çevrim tablosundan okunmakta iken, Denklem (4.10)'da mikro rotasyon açı değerleri 2'nin kuvvetleri ile ifade edilmektedir. Öte yandan her iki denklemde öteleme ve toplama operasyonları mevcuttur. Her bir iterasyon adımında iterasyon sayısına bağımlı olan K ölçek faktörü Denklem (2.19)'da mevcut iken, Denklem (4.10)'da K ölçek faktörü bulunmamaktadır. Ancak "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" [9] algoritmasında sadece belirli alanlarda (açı bölgesi, domain) iterasyon sayısından bağımsız K ölçek faktörü $\left(1 \; veya \; 1/\sqrt{2}\right)$ kullanılmaktadır. İleriki bölümlerde [9] için ölçek faktörünün rotasyon denkleminde kullanımı detaylı bir şekilde açıklanacaktır.

"Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" [9] algoritmasında vektör rotasyonlarının tek yönlü oluşu ve K ölçekleme faktörünün iterasyon sayısından

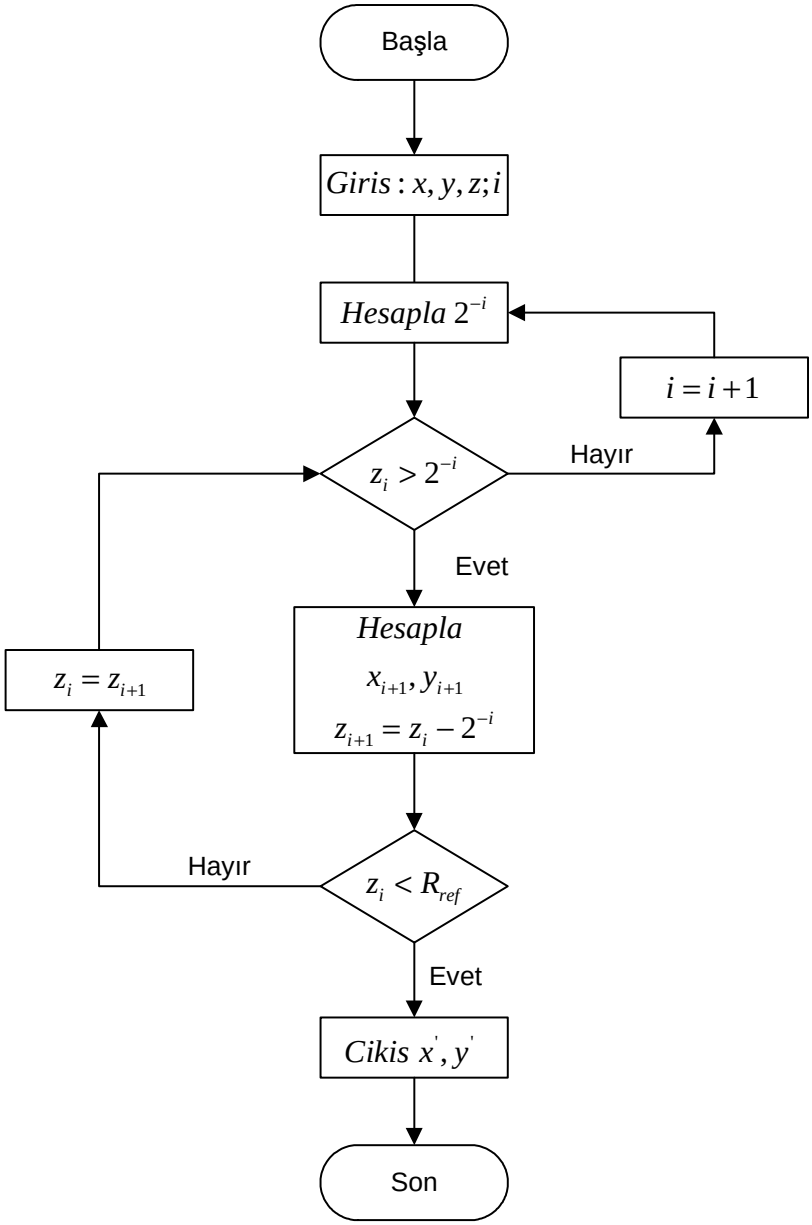
bağımsız oluşu, algoritmanın daha az iterasyonda sonucu elde edeceği anlamını çıkarmaktadır. Bu durum ise Klasik CORDIC algoritmasına [1] göre daha az operasyon kullanımı ve düşük güç tüketimi demektir.

CORDIC Rotasyonel Denklemi	**Denklem**	**Operasyon Sayısı (Bir iterasyon adımı için)**	**Ölçek Faktörü**	**Vektör Rotasyonu**	**Çevrim Tablosu Kullanımı**
Klasik CORDIC[1]	$\begin{bmatrix} x_{i+1} \\ y_{i+!} \end{bmatrix} = K.\begin{bmatrix} 1 & r_i.2^{-i} \\ -r_i.2^{-i} & 1 \end{bmatrix}.\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ $z_{i+1} = z_i - r_i.\arctan(2^{-i})$ $(K = \prod_{i=0}^{n-1} \cos(\tan^{-1}(2^{-i}))$ olmak uzere)	2 adet öteleme ve 2 adet toplama/ çıkarma	Var	İki yönlü	Var
Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC [9]	$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1-2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1-2^{-(2i+1)} \end{bmatrix}.\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ $z_{i+1} = z_i - r_i.2^{-i}$	4 adet öteleme ve 4 adet toplama/ çıkarma	Yok	Tek yönlü	Yok

Tablo 4 - 3. CORDIC rotasyonel denklemlerinin karşılaştırılması

$K = A_n = \prod_{i=0}^{n} \sqrt{1+2^{-2i}}$ Ölçek faktörünün iterasyon sayısına bağlı oluşu Klasik CORDIC rotasyonel denkleminin dezavantajlı yönlerinden birisidir. İterasyon sayısı arttıkça ölçek faktörü de artacak, dolayısıyla algoritma ilave toplama/çıkarma ve öteleme operasyonlarına ihtiyaç duyacaktır. Bu durum donanımda kullanılan alanı artıracaktır. Ayrıca Klasik CORDIC algoritmasında [1] vektör rotasyonları iki yönlü olarak yürütüldüğü için θ hedef açısına, tek yönlü vektör sayısına oranla daha fazla mikro rotasyon adımı sonunda ulaşılacaktır.

Denklem (4.10)'da ifade edilen CORDIC rotasyon denklemine ait akış diyagramı Şekil 4-2'de verilmiştir.



Şekil 4 - 2. İterasyon basamaklarının uygun seçim işlemi (iterasyon seçim akış diyagramı)

4.2.1.1. İterasyon Seçim Algoritmasının Çalışma Şekli:

Şekil 4-2'de verilen seçim algoritmasının çalışma şekli aşağıda açıklanmıştır.

Algoritma x, y, z ve i başlangıç atamalarının yapılması ile yürütülmeye başlamaktadır.

İterasyon seçim algoritmasında her bir iterasyon basamağının başlangıcında, residue açı z_i, i. temel rotasyon açısı değeri olan 2^{-i} ile karşılaştırılır.

Eğer $z_i < 2^{-i}$ koşulu mevcut ise i iterasyon sayısı bir artırılır ve $z_i \geq 2^{-i}$ koşulu oluşuncaya kadar bu işlem tekrarlanır.

Eğer $z_i \geq 2^{-i}$ koşulu mevcut ise (x_{i+1}, y_{i+1}) değerleri ve z_{i+1} residue açısının yeni değeri Denklem (4.10) 'a göre hesaplanır.

Bu işlem kullanıcının tanımladığı R_{ref} kesinliğine ulaşılıncaya kadar tekrarlanır. $z_i < Rref$ koşulu gerçekleştiği andaki x ve y vektör bileşenleri sırasıyla $\cos(\theta)$ ve $\sin(\theta)$ sonuçlarını üretmektedir.

Bu algoritma tek yönlü vektör rotasyonları ile yürütüldüğü için seçilen iterasyon adımlarının sayısı algoritmanın yürütülmesi için ihtiyaç duyulacak olan iterasyon sayısını verecektir. Bu sebeple algoritma sonlandığı andaki i sayacının değeri yürütülen toplam iterasyon sayısını verecektir.

4.2.1.2. Dezavantajları

"Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" [9] algoritması, Klasik CORDIC'deki [1] ölçek faktörünü iterasyon sayısından bağımsız olarak kısmen kaldırmakla birlikte Klasik CORDIC algoritmasına yakın doğrulukta sonuçlar elde edilir. Bu metodun en büyük sakıncası yakınsama değerlerinin aşırı derecede dar oluşudur. Yakınsama değerlerinin dar oluşu algoritmanın yüksek iterasyon sayısında sonuçlara ulaşması anlamına gelmektedir ki, bu nedenle "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" algoritması [9] pratikte kullanılabilir değildir. Ancak [10], [11] ve [12]'de açıklanan CORDIC yöntemlerine temel oluşturması bakımından "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" algoritması [9] oldukça önemli bir çıkış noktasıdır.

4.2.2. Modifiye Edilmiş Gerçekte Ölçekleme Bağımsız Adaptif CORDIC Dönüştürücü Algoritması

[9]'da açıklanan CORDIC algoritmasının bahsettiğimiz dezavantajlarını ortadan kaldırmak ve algoritmayı uygulanabilir bir hale getirmek amacıyla; daha etkin bir yaklaşım [10]'da öne sürülmüştür. Bu yaklaşım [9]'un mantığını temel almıştır. Bununla birlikte [9]'da ifade edilen mevcut tasarım modifiye edilerek algoritma uygulamalarda kullanılabilecek performansa getirilmiştir.

Bu yeni yaklaşımda önceki algoritmanın en önemli özelliği olan ölçekleme bağımsız (scaling-free) özelliğini korunarak yeni açı hesaplama oranları ile eski algoritma modifiye edilmiştir. Koushik'in yeni önerdiği modifiye edilmiş bu yaklaşımda iterasyon sayısından bağımsız K ölçek faktörü sadece θ hedef açı değerine bağlıdır. K ölçek faktörü θ hedef açısının kartezyen koordinat düzleminde bulunduğu alana (domain) göre $\left(1 \text{ veya } 1/\sqrt{2}\right)$ değerlerini alabilmektedir.

Bu algoritmanın geliştirilmesindeki ana düşünce "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" algoritmasının [9] genel yapısına sahip çıkılarak, tüm koordinat uzayındaki yakınsama oranlarının genişletilmesidir. Bu şekilde algoritma uygulanabilir bir hal almıştır. Algoritma, sadece gerekli temel rotasyon adımlarında yürütülen bir mimari tasarıma sahiptir. Bu adaptif mimari [10]; "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" algoritmasının uygulanabilir bir formudur.

Bu yeni CORDIC algoritması [10], Klasik CORDIC [1] ile maksimum 16 iterasyon için karşılaştırıldığında;

- Aynı doğruluk oranlarında yaklaşık %50 daha az iterasyonda sonuca ulaşmaktadır.
- Aynı sayıda tam toplayıcı (full adder) kullanmaktadır, (Klasik: 768)
- %30.6 daha az register kullanmaktadır. (Klasik: 768, Önerilen: 533)

4.2.2.1. Modifiye Edilmiş Gerçekte Ölçekleme Bağımsız Adaptif CORDIC Dönüştürücü Algoritması Rotasyonel Denklemlerinin Gelişimi

Koushik'in bu yeni yaklaşımında [10] daha az toplama/çıkarma ve öteleme birimi kullanılmaktadır.

(4.6) nolu denklem i iterasyon indeksinde, (4.12) nolu denklemdeki sınırlamayı zorunlu kılar.

b bit sayısı işlemcinin üzerinde işlem yapabildiği en uzun verinin bit sayısı (kelime uzunluğu) olmak üzere;

$$\frac{(b-2.585)}{3} \le i \le b-1 \tag{4.11}$$

$$\frac{(b-2.585)}{3} = p \le i \le b-1 \tag{4.12}$$

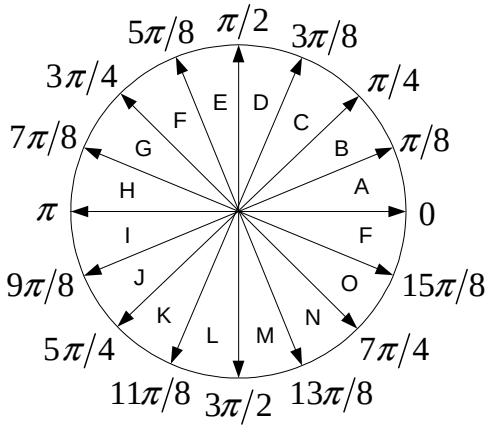
"Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" [9] rotasyonel denklemi olan Denklem (4.10), $(b-p)$ sayıda iterasyon için yeniden düzenlenecek olursa "Modifiye Edilmiş Gerçekte Ölçekleme Bağımsız Adaptif CORDIC Dönüştürücü" algoritması [10] rotasyonel denkleminin en genel hali elde edilir.

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \prod_{i=p}^{b-1} \begin{bmatrix} 1-2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1-2^{-(2i+1)} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.13}$$

$$z_{i+1} = z_i - r_i.2^{-i}$$

4.2.2.2. Bu algoritmanın modifiye edilmesinde sırasıyla yapılanlar:

- [9]'un yakınsama oranları genişletilmiştir.
- Bu amaçla değişken azaltım (argument reduction) tekniği kullanılarak hesaplanacak toplam açı oranı azaltılmıştır. (alan katlama tekniği) Sonuçta yakınsama oranları zenginleştirilmiştir.
- Değişken azaltım tekniği kullanılarak küçük ϕ acıları ile bağlantılı büyük θ hedef açısına ulaşılabilmektedir.
- Bu amaçla; Koordinat uzayında 4 adet quadrantın her biri 4 eşit alana bölünür. Böylece koordinat uzayı 16 eşit alana parçalanmış olur. { 22.5 derece x 16 = 360 derece } B, C ve D alanları, A alanına göre geri katlanmış durumdadır. Bu sebeple kullanılan teknik "alan katlama tekniği" (domain folding) olarak adlandırılmaktadır.



Şekil 4 - 3. Koordinat uzayının 16 eşit alana parçalanması

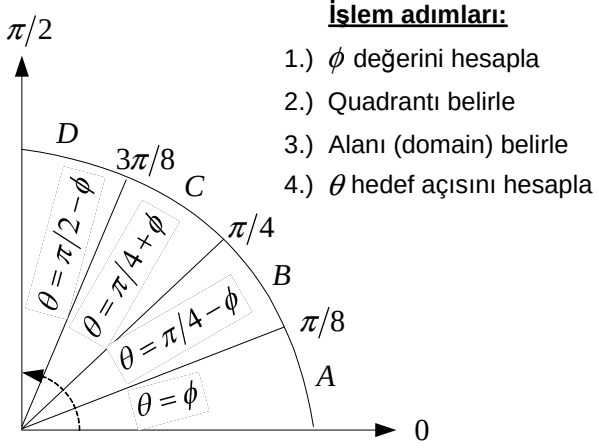
İlk quadrant'daki θ hedef açısının yayılımını inceleyecek olursak, θ'nın A, B, C ve D alanları boyunca uzandığını görürüz. Her alan $[0,\pi/8]$ aralığında sınırlı kalmak üzere ϕ açısı ile yeniden tanımlanabilir.

$$\theta = \phi \qquad Alan\ A \tag{4.14}$$

$$\theta = \frac{\pi}{4} - \phi \qquad Alan\ B \tag{4.15}$$

$$\theta = \frac{\pi}{4} + \phi \qquad Alan\ C \tag{4.16}$$

$$\theta = \frac{\pi}{2} - \phi \qquad Alan\ D \tag{4.17}$$



Şekil 4 - 4. İlk quadrantın dört alana parçalanması ve hedef açısının belirlenmesi

Gelişi güzel bir θ hedef açısına sahip başlangıç vektörünün rotasyon sonrasında elde edilecek yeni vektör koordinatlarının (x', y') hesaplanmasında;

İlk adım: Hedef açısının sırasıyla hangi quadrant ve alanda (domain) bulunduğunun tespit edilmesidir.

İkinci adım: Hedef açısının bulunduğu açısal bölge tespit edildikten sonra (4.22)-(4.25) arasındaki denklemlerden uygun olanları uygulanarak hesaplama gerçekleştirilmektedir.

(4.14) ve (4.17) arasındaki denklemleri kullanarak (x', y') koordinat değerleri farklı alanlar için ϕ açısı cinsinden ifade edilebilirler. (4.18) ve (4.21) arasındaki denklemler bu durumu açıklamaktadır.

$$\begin{bmatrix} x_{fA} \\ y_{fA} \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}, \; Alan\ A \tag{4.18}$$

$$\begin{bmatrix} x_{fB} \\ y_{fB} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} (\cos\phi + \sin\phi) & (\cos\phi - \sin\phi) \\ -(\cos\phi - \sin\phi) & (\cos\phi + \sin(\phi) \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}, \; Alan\ B \tag{4.19}$$

$$\begin{bmatrix} x_{fC} \\ y_{fC} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} (\cos\phi - \sin\phi) & (\cos\phi + \sin\phi) \\ -(\cos\phi + \sin\phi) & (\cos\phi - \sin(\phi) \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}, \; Alan\ C \tag{4.20}$$

$$\begin{bmatrix} x_{fD} \\ y_{fD} \end{bmatrix} = \begin{bmatrix} \sin\phi & \cos\phi \\ -\cos\phi & \sin\phi \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}, \; Alan\ D \tag{4.21}$$

x_{f*}, * tarafından belirlenen alanda uzanan hedef açısına sahip CORDIC rotasyonundaki sonuç vektörünü temsil eder.

(4.18)-(4.21) aralığındaki denklemler yeniden düzenlenerek (4.22)-(4.25) aralığındaki denklemler elde edilebilir.

$[x_{1+} \quad y_{1+}]^T$ ve $[x_{1-} \quad y_{1-}]^T$ sonuç vektörleri sırasıyla $+\phi$ ve $-\phi$ açıları için CORDIC rotasyonunu gösterir. Bu duruma açıklık getirecek olursak (4.22) ve (4.25) arasındaki denklemlerdeki '+' ve '–' işaretleri sırasıyla pozitif ve negatif CORDIC rotasyonlarını temsil etmektedir.

$$x_{fA} = x_{1-} \;,\; y_{fA} = y_{1-} \;,\; \phi = \theta \tag{4.22}$$

$$x_{fB} = \frac{1}{\sqrt{2}} . [x_{1+} \quad + y_{1+}] \;,\; y_{fB} = \frac{1}{\sqrt{2}} . [-x_{1+} \quad + y_{1+}] \;,\; \phi = \frac{\pi}{4} - \theta \tag{4.23}$$

$$x_{fC} = \frac{1}{\sqrt{2}}.[x_{1-} \quad + y_{1-}] \quad , \quad y_{fC} = \frac{1}{\sqrt{2}}.[-x_{1-} \quad + y_{1-}] \quad , \; \phi = \theta - \frac{\pi}{4} \tag{4.24}$$

$$x_{fD} = y_{1+} \; , \; y_{fD} = -x_{1+} \; , \quad \phi = \frac{\pi}{2} - \theta \tag{4.25}$$

(4.22)-(4.25) arasındaki denklemler kullanılarak ilk quadrant'ın herhangi bir alanı üzerinde uzanan θ hedef açısı hesaplanabilir. $[0, \pi/8]$ aralığında ilk quadrant'daki θ hedef açısının yayılımını inceleyecek olursak, θ'nın A, B, C ve D alanları boyunca uzandığını görürüz. Her alan $[0, \pi/8]$ aralığında sınırlı kalmak üzere ϕ açısı ile CORDIC rotasyon sonuçları hesaplanabilir.

Buradaki en önemli nokta: B veya C alanlarında bulunan sabit ölçek faktörü $1/\sqrt{2}$'nin iterasyon sayısından bağımsız oluşudur. Yani ölçek faktörü oluşan her temel rotasyondan bağımsızdır. Diğer bir taraftan A ve D alanları ölçek faktörüne ihtiyaç duymazlar. Şekil 4-5'de Klasik CORDIC'de iterasyon sayısına bağımlı ölçek faktörü kullanımı Koushik'in "ölçekleme bağımsız" (scaling-free) yaklaşımı ile ortadan kalkmıştır.

$$A_n = \prod_{i=0}^{n} \sqrt{(1 + 2^{-2i})}$$

Şekil 4 - 5. İterasyon sayısına bağımlı ölçek faktörü kullanımı

(4.22)-(4.25) arasındaki denklemler göstermektedir ki belirli bir hedef açısı için rotasyon sadece bir yönde (+ ya da -) olmalıdır.

Bundan dolayı "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" algoritması [9] için geçerli olan "vektör rotasyonu sadece tek bir yönde ardışıl rotasyonlardan oluşmalı" özelliği bu yöntemde de halen sağlanmaktadır. Şekil 4-6'da algoritmada kullanılan alan katlama tekniğine ait sözde kod (pseudo code) verilmiştir.

```
if (x ≥ y) then
    if (x ≥ x_AB) then                Alan A
        x' = x
        y' = y
    else                              Alan B
        x' = (x + y)/√2
        y' = −(y − x)/√2)
    endif
else
    if (x < x_CD) then                Alan D
        x' = y
        y' = x
    else                              Alan C
        x' = (x + y)/√2
        y' = (y − x)/√2)
    endif
endif
```

Şekil 4 - 6. Alan katlama tekniği için sözde kod

Tablo 4-4'de $\theta \in [0, 2\pi]$ aralığında dönüştürülen hedef açısı, $\phi \in [0, \pi/8]$ aralığındaki yakınsama açısı cinsinden ifade edilmiştir.

ALAN	θ	**ALAN**	θ	**ALAN**	θ	**ALAN**	θ
A	ϕ	**E**	$(\pi/2)+\phi$	**I**	$\pi+\phi$	**M**	$(3\pi/2)+\phi$
B	$(\pi/4)-\phi$	**F**	$(3\pi/4)-\phi$	**J**	$(5\pi/4)-\phi$	**N**	$(7\pi/4)-\phi$
C	$(\pi/4)+\phi$	**G**	$(3\pi/4)+\phi$	**K**	$(5\pi/4)+\phi$	**O**	$(7\pi/4)+\phi$
D	$(\pi/2)-\phi$	**H**	$\pi-\phi$	**L**	$(3\pi/2)-\phi$	**P**	$2\pi-\phi$

Tablo 4 - 4. Koordinat uzayında hedef açısı dönüşümü

Hedef açısının bulunduğu quadrant'a bağlı olarak sonuç vektör bileşenlerinin işareti ve (x, y) atamaları değişmektedir. Bu durum Tablo 4-5'de ifade edilmiştir. Koordinat ekseninin simetri özelliğinden yararlanılarak herhangi bir quadrant'daki hedef açıları alan katlama tekniği ile elde edilebilir.

1. Quadrant $(0, \pi/2)$			2. Quadrant $(\pi/2, \pi)$			3. Quadrant $(\pi, 3\pi/2)$			4. Quadrant $(3\pi/2, 2\pi)$		
Alan	x_f	y_f	**Alan**	x_f	y_f	**Alan**	x_f	y_f	**Alan**	x_f	y_f
A	x_{fA}	y_{fA}	**E**	y_{fA}	$-x_{fA}$	**I**	$-y_{fA}$	$-x_A$	**M**	x_{fA}	$-y_{fA}$
B	x_{fB}	y_{fB}	**F**	y_{fB}	$-x_{fB}$	**J**	$-x_B$	$-y_{fB}$	**N**	$-y_{fB}$	x_{fB}
C	x_{fC}	y_{fC}	**G**	y_{fC}	$-x_{fC}$	**K**	$-x_C$	$-y_{fC}$	**O**	$-y_{fC}$	x_{fC}
D	x_{fD}	y_{fD}	**H**	y_{fD}	$-x_{fD}$	**L**	$-y_{fD}$	$-x_D$	**P**	x_{fD}	$-y_{fD}$

Tablo 4 - 5. Alan katlama tekniği kullanılarak farklı quadrantlar için vektör bileşenleri ve işaretleri

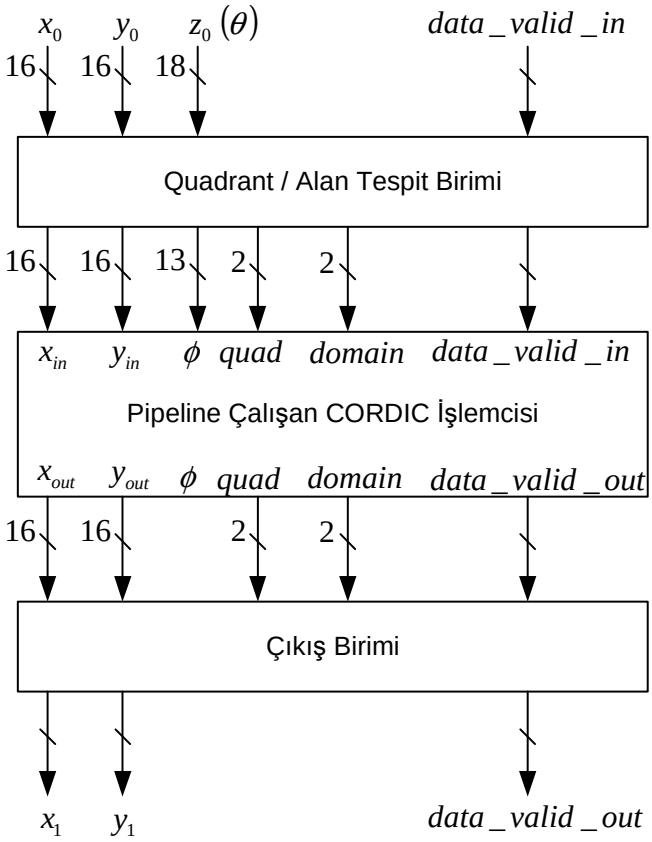
Koushik bu yeni yaklaşımında [10] yakınsama değerlerini genişletmek için bir yöntem geliştirmiştir. Bu yaklaşımda iterasyon basamaklarının birden fazla tekrarlanması düşünülmüştür. Yakınsama oranlarını artırmak için iterasyonlar tekrarlama döngüsüne sokulmaktadır.

Tek yönlü vektör rotasyonları ile algoritmanın yürütülüyor olması hesaplama karmaşıklığını ciddi oranda azaltmaktadır. Ancak Koushik 20-bit ve üzeri giriş verileri için önerdiği mimarinin pratikte uygulanabilir olmadığını [10]'da ifade etmiştir. 16 bitlik giriş verileri için CORDIC rotasyonu yürütülmesine ilişkin örnek bir pipeline mimari takip eden bölümde tanıtılacaktır.

4.2.2.2.1 Modifiye Edilmiş Gerçekte Ölçekleme Bağımsız Adaptif CORDIC Dönüştürücü Algoritmasının Mimarisi:

Tasarım üç temel birimden oluşur.

1. **Quadrant / Alan Tespit Birimi**
2. **Pipeline Çalışan CORDIC İşlemcisi** ($\phi = [0, \pi/8]$ yakınsama oranına sahip)
3. **Çıkış Birimi**



Şekil 4 - 7. Mimari Birimler

Algoritmanın Şekil 4-7'de gösterilen üç temel birimini açıklayacak olursak:

1. **Quadrant / Alan Testpit Birimi:**

x_0 ve y_0 girişleri 16-bit genişliğindedir. z_0 girişi 18-bit genişliğindedir.

En büyük açı olan θ hedef açısı, $[0,2\pi]$ aralığında uzanan z_0 girişi olarak atanır.

Quadrant / Alan Tespit Birimi'nde ilk yapılması gerekli iş; hedef açısı θ'nın sırasıyla quadrant ve alan bilgilerini testpit etmektir.

Ardından 13 bitlik modifiye edilmiş ϕ yakınsama açısını üretmek için alan katlama tekniği uygulanır.

Quadrant /Alan Tespit Birimi çıkışında 2 bitlik 2 sinyal üretilmektedir. Bu iki bitlik sinyallerin adı; "quad" ve "alan" dır.

Quad sinyali: θ hedef açısının kartezyen koordinat düzleminde hangi quadrantda bulunduğunu belirtir.

Domain sinyali: İlk quadrantda geri katlanmış durumdaki (1. quadrantdaki) uygun alanı belirtir.

Bu birimde tüm operasyon 1 saat çevriminde (clock cycle) yürütülmektedir.

2. Pipeline Çalışan CORDIC İşlemcisi:

Bu tasarımda 16-bit fixed point algoritması göz önüne alınmıştır. Başlangıç vektörünün quadrant ve alan hakkındaki veri bilgisi "Quadrant/Alan Tespit" biriminde üretilip lokal register transfer yapısıyla bir sonraki birim olan "Pipeline Çalışan CORDIC İşlemcisi" birimine transfer edilir.

CORDIC temel mikro rotasyonları "Pipeline Çalışan CORDIC İşlemcisi" biriminde yürütülmektedir. [10]'da ulaşılabilir maksimum hedef açısı ± 7.16 derecedir. Bu maksimum değere i=4'den i=15'e kadar (p=4) iterasyonları ile ulaşılabilmektedir. Yakınsama aralığının dar oluşu bir dezavantajdır. Fakat bu dezavantaj, alan katlama tekniği kullanılarak ± 22.5 dereceye yükseltilebilir. Sonuç itibariyle pipeline birim, 16-bitlik (b=16) dahili kelime uzunluğuna ve $\phi = [0, \pi/8]$ yakınsama oranına sahiptir.

Pipeline yapıda 7. rotasyon adımından itibaren rotasyon adımları birleştirilmiş olup, bir pipeline rotasyon adımında iki rotasyon adımı aynı anda yürütülecektir. Çünkü 7. rotasyon adımından itibaren 2i+1 öteleme operasyonları göz önüne alınmamakta yani işlem sayısı azalmaktadır. Bu da işlem süresinin azalması demektir. Ulaşılabilecek maksimum hedef açısı için her temel rotasyon birimi $i = 4$ üst sınır değeri ile sınırlandırılmıştır ki maksimum açı $22.5^0 (\pi/8)$ olmaktadır. Aradaki yakınsama açı farkı $22.5^0 - 7.16^0 = 15.34^0$, $i = 4$ indeksi için gerekli temel rotasyon birimlerinin sayısını göstermektedir. Tablo 4-6'da bu durum ifade edilmiştir. $i = 4$ indisi için bir ve birden fazla rotasyon için kullanılmakta iken $i = 5, 6, \cdots, 15$ indisleri için sadece bir temel rotasyon birimi gereklidir.

$[0, \pi/8]$ yakınsama oranının sağlanması için $i = 4$ rotasyon adımı 6 kez ve $i = 5, 6, ...14$ rotasyon adımlarının her biri bir kez kullanılmaktadır. Yani mimari yapı 6 adet i=4 ve 11 adet iterasyon (i=5,6..,15) olmak üzere toplam 17 rotasyon

adımından oluşmaktadır. 16 bit uygulamada en büyük sağa öteleme değeri 4 olabilir. Bu durum Klasik CORDIC yaklaşımı [1] ile karşılaştırıldığında bir dezavantaj gibi görünür. Çünkü Klasik CORDIC [1] algoritması 16 bit veri uzunluğu işlemler için 16 rotasyon adımına ihtiyaç duyarken, Koushik'in yaklaşımında aynı veri uzunlığunda işlemler için CORDIC algoritması [10] görünürde 17 rotasyon adımına ihtiyaç duyar. Ancak rotasyon adımlarının tamamının aktif olması durumu mümkün değildir. Ayrıca 17 rotasyon adımının tamamı asla kullanılmamaktadır. Yakınsama açısı olan $\pi/8$ değerine en yakın büyüklükte olan açı 1011111111111 olduğu durumda giriş vektörünün rotasyonu için 15 adet rotasyon adımının aktif edilmesi gerekir. Eğer rotasyon adımlarının tamamı aktif olsaydı 22.5 derecelik yakınsama açısı değeri aşılmış olacaktı. Bu durumda hedef açısı Denklem (4.26)'de tanımlandığı gibi hesaplanırdı ki bu değer müsade edilen oranın dışında kalmaktadır.

$$\theta = 6.2^{-4} + \sum_{i=5}^{15} 2^{-i} = 25.06^{0} > 22.5^{0} \tag{4.26}$$

$i = (7,8),(9,10),(11,12)$ rotasyon adımlarının birleştirilmesi sonucunda (6 adet i=4), 5, 6, (7-8), (9-10), (11-12), (13-14) olmak üzere pipeline rotasyon adım sayısı 16'dan 12'ye düşürülmüştür.

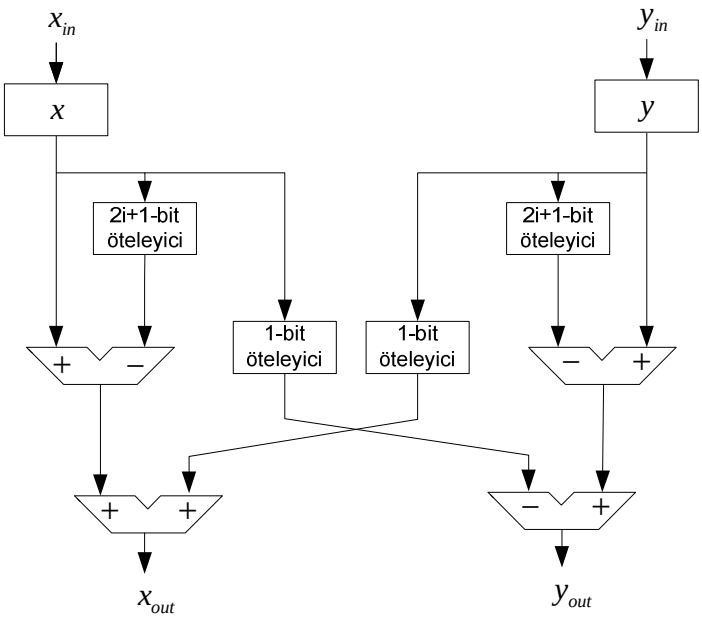
"Quadrant / Alan Tespit" birimi ve "Çıkış" birimi birer rotasyon adımında işlem gördüklerinden dolayı sonuç değerlerine, en büyük açı değeri için mimari yapı toplam 14 rotasyon adımı uzunluğunda pipeline yapıdan oluşmaktadır. Sonuç itibariyle işlemcinin gecikme süresi 14 clock cycle'dır. Koushik'in yeni yaklaşımındaki [10] tasarım bileşenleri için yürütülen rotasyon adım sayısı Tablo 4-6'da gösterilmiştir.

BİRİMLER	**Rotasyon Adım Sayısı**
Quadrant / Alan Testpit Birimi	1
Pipeline Çalışan CORDIC İşlemcisi	12
Çıkış Birimi	1

Tablo 4 - 6. Mimari birimlerde yürütülen rotasyon adımları

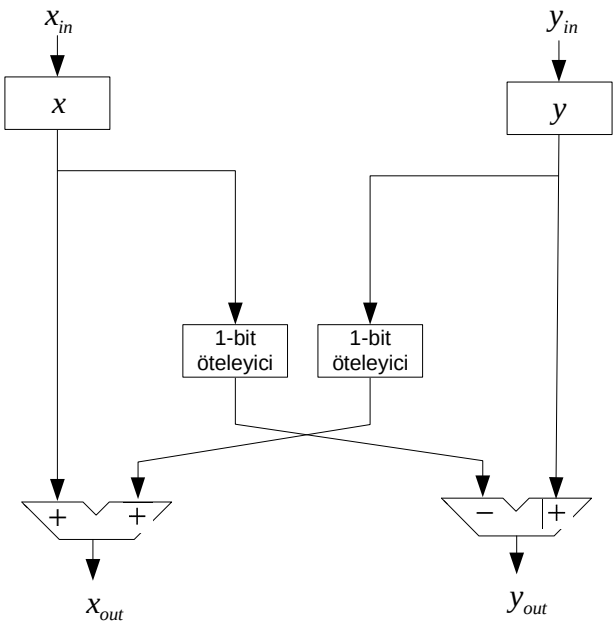
İki tip modified temel rotasyon birimi vardır. Şekil 4-8 ve Şekil 4-9'da her iki tip rotasyon mimarisi görülmektedir.

b=16 bit uygulamalar için; temel rotasyonlar $p < 2i+1 < b$ durumu için (yani $4 \le i \le 7$) her mikro rotasyon 4 adet toplama/çıkarma ve 4 adet öteleme birimine eşdeğerdir. Algoritmanın basitleştirilmiş blok diyagramı Şekil 4-8'de verilmiştir.



Şekil 4 - 8. Modified temel rotasyon birimi blok diyagramı (i=4,5,6,7) (Tip-1)

$i \ge b/2$ yani $i \ge 8$ olduğunda her mikro rotasyon 2 adet toplama/çıkarma ve 2 adet öteleme birimine eşdeğerdir. Bu durumda (2i+1) girişli sağa öteleme işlemleri ihmal edilebilir. ($i = 8,9,...14$ nolu rotasyon adımlarında) Algoritmanın basitleştirilmiş blok diyagramı Şekil 4-9'da verilmiştir.



Şekil 4 - 9. Modified temel rotasyon birimi blok diyagramı (i=8,9,…,15) (Tip-2)

Bununla birlikte temel sorun hala devam etmektedir. Verilen θ hedef açısı için gerekli rotasyon birimlerinin nasıl seçileceği önemlidir. Bunun için giriş verisini ikilik sistemde 2'nin negatif kuvvetleri biçiminde ifade edeceğiz.

işaret	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}		
16	**15**	**14**	**13**	**12**	**11**	**10**	**9**		
→	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
	8	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Tablo 4 - 7. Giriş verisinin İkinin negatif kuvvetleri biçiminde gösterimi

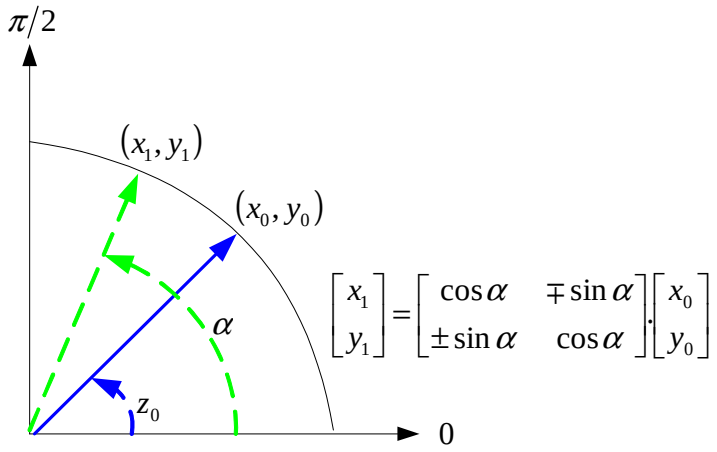
				MSB3	**MSB2**	**MSB1**		
işaret	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
16	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
								LSB
→	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Tablo 4 - 8. 16 bit veri uzunluğunda giriş verisinini (0, 2pi) aralığında 2'nin negatif kuvvetleri cinsinden ikili sayı siteminde gösterimi

Tablo 4-8'de hedef açı için temel rotasyon rotasyon adımının seçimi 13-bitlik ϕ açısındaki uygun lojik-1 pozisyonlarının seçimi ile gerçekleşir. Lojik-1 girişi

olduğunda uygun temel rotasyon aktif hale gelir. 3 adet en önemli bit (MSB) $\left(2^{-2}, 2^{-3}, 2^{-4}\right)$, $i = 4$ olan aktif rotasyon adım sayısına karar vermek için kullanılır.

Bu amaçla uygun olan MSB'lere lojik-1 değerler yüklenir. Önemli bir durum olan $\pi/8$ için uygun MSB değerleri sırasıyla 110 ikili değerini alır. $\{(110)_2 = 6\}$ Bu durumda i=4 olan 6 adet rotasyon adımı aktif edilmiştir. Yani 3 adet MSB'in oluşturduğu ikili ifadenin onluk sayı sistemine karşılık gelen değeri, i=4 olan aktif rotasyon adımı sayısını belirler. Kombinasyonel kod çözücü devresi, bu operasyonu çözmek kullanılır. Geriye kalan $\left[2^{-5}, 2^{-14}\right]$ arasındaki bitlerden uygun olanları temel rotasyon birimininin seçimi için kullanılır. 5. ve 14. rotasyon adımları arasında bir bit iterasyona karşılık gelen vektör rotasyonu Şekil 4-10'da ifade edilmiştir.



Şekil 4 - 10. Bir iterasyon adımına karşılık gelen vektör rotasyonu

Mimariye açıklık getirmek için 16-bit CORDIC rotasyonunun yürütülmesi ile ilgili bir örnek verecek olursak;

ÖRNEK-2: Mimari tasarımı için onluk sayı sisteminde 1 değerini ikilik sayı sisteminde 0 100 000 000 000 000 varsayıyoruz. n-bit sayısının değerine göre farklı varsayımlar da ifade edilebilir. Burada n=16-bit varsayımı yapılmıştır ki algoritma en iyi performansı n=16 bit için vermektedir.

Koushik'in yeni CORDIC yaklaşımında [10] alan katlama tekniği kullanılarak koordinat uzayında uzanan tüm açılar $[0, \pi/8]$ oranında etkin olarak haritalanmıştır.

Etkin maksimum hedef açısı için $\pi/8 = 0.392699$ olacaktır.

Mimariye açıklık getirmek için 16-bit CORDIC rotasyonunun yürütülmesi örneğini ikili sayı sisteminde Şekil 4-11'deki gibi gösterecek olursak;

$$\begin{array}{lcll} & & 2^0 & \\ 1 & = & 0 \quad 100000000000000 & = \quad 2^0 \\ \pi/8 & = & 0 \quad 001100100100010 & = \quad 0.392699 = 2^{-2}+2^{-3}+2^{-6}+2^{-9}+2^{-13} \\ 2\pi & = & 0\,11 \quad 001001000011111 & = \quad 6.283185 \\ & & & = \quad 2^2+2^1+2^{-2}+2^{-5}+2^{-10}+2^{-11}+2^{-12}+2^{-13}+2^{-14} \end{array}$$

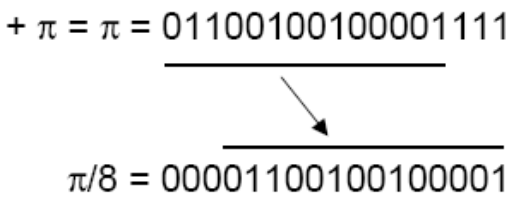
Şekil 4 - 11. Yakınsama açısının ikili sayı sisteminde gösterimi

'1' varsayımı kullanılarak $\pi/8$ açısı ikili sayı sisteminde $0\,001100\,100\,100\,010$ olarak $2^{-16} \cong 0$ hata oranı ile temsil edilir.

Modifiye edilmiş yakınsamada herhangi bir mutlak açı değerinin temsil edilmesinde en önemli ilk 3 bit (MSB- Most Significant Bit) ihmal edilebilir ve en az öneme sahip 13 bit (LSB-Least Significant Bit) kullanılabilir (16-bit veri yolu uzunluğu için). Mimaridekі; açı yakınsama işlemindeki hesaplama miktarını azaltmak için ifade edilen bu özellikten faydalanılır.

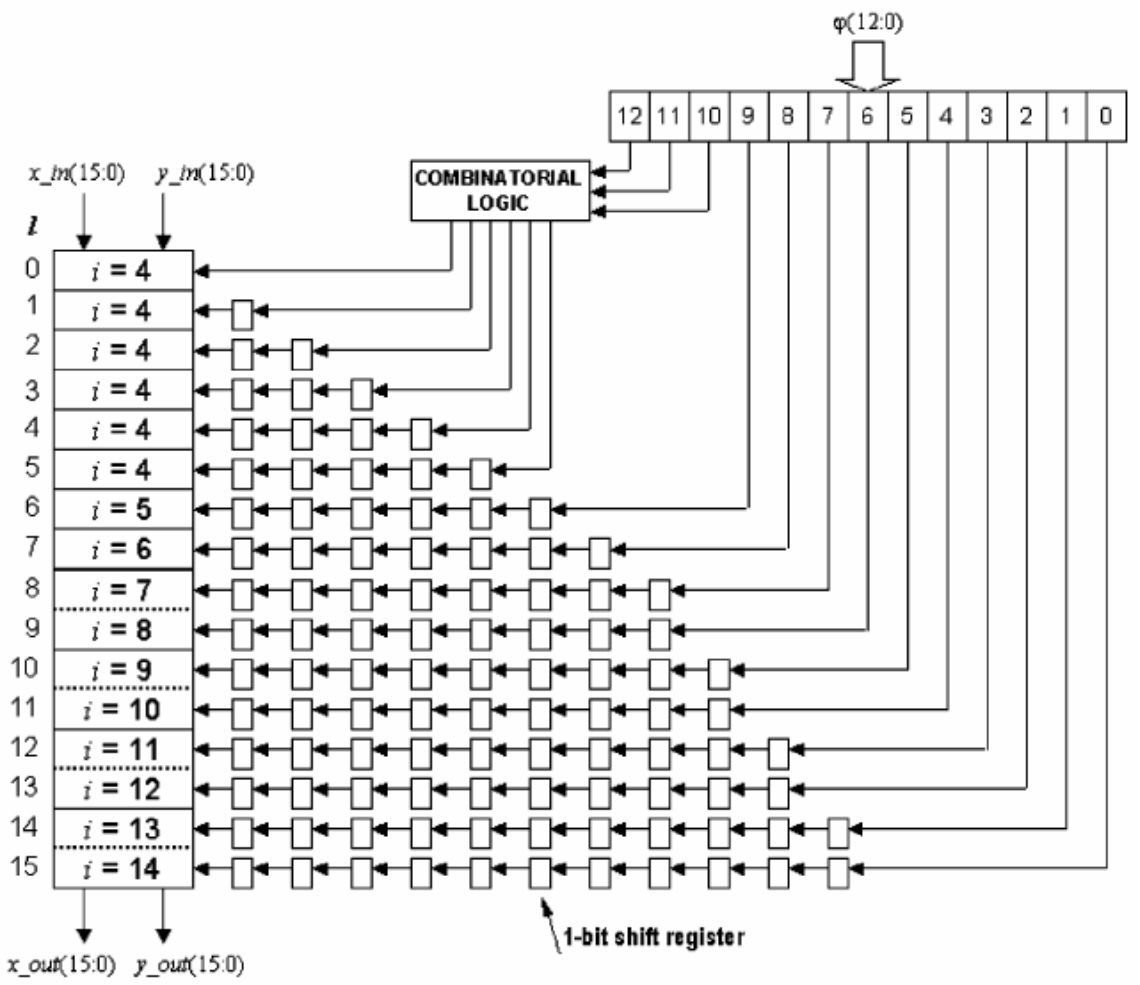
Maksimum hedef açı için müsaade edilen değer $\pi/8$'dir.

($\pi/8$) değerine: π'nin ikili sayı sistemindeki karşılığı olan değerin 3 adet sağa ötelenmesi işlemi sonucunda ulaşılır. Bu durum Şekil 4-12'de gösterilmiştir.



Şekil 4 - 12. Maksimum hedef açısına yakınsama - bit gösterimi

6 adet $i = 4$ rotasyon adımında uygun olanları seçmek için girişleri ϕ açısının $10.,11.,12.$ bit pozisyonları için kombinasyonel bir devre kullanılmaktadır. Bu devre yapısı Şekil 4-13'de gösterilmiştir.

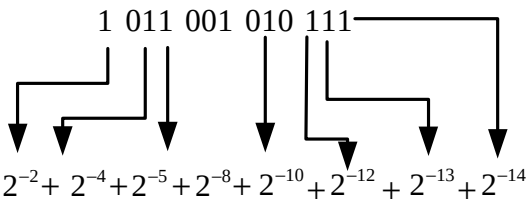


Şekil 4 - 13. “Modifiye Edilmiş Gerçekte Ölçekleme Bağımsız Adaptif CORDIC Dönüştürücü” algoritması pipeline mimarisi

“Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC” algoritmasında [9] olduğu gibi hedef açısına yakınsama vektörün aynı yönde rotasyonu ile gerçekleşir. Bundan dolayı hedef açı basit olarak 2^{-i} ötelemelerinin toplamının yakınsamasıdır.

Hedef açıya ulaşmada her rotasyonel adımının kullanılmayıp sadece uygun olanların kullanılması aritmetik işlemleri azaltmaktadır.

ÖRNEK-3: $\phi = 20^0 = 0.349\ rad.$ açısının ikilik sayı sisteminde işaretsiz gösterimi Şekil 4-14’deki gibi olacaktır.



Şekil 4 - 14. 20 derecelik açı için lojik bit dizisi

ϕ hedef açısına ulaşabilmek için $i = 4, 4, 4, 4, 5, 8, 10, 12, 13, 14$ rotasyon adımları aktif edilmelidir. İlgili rotasyon adımını aktif yapmak için uygun bit pozisyonuna lojik-1 girilirken, rotasyon adımını inaktif yapmak için uygun bit pozisyonuna lojik-0 girilir.

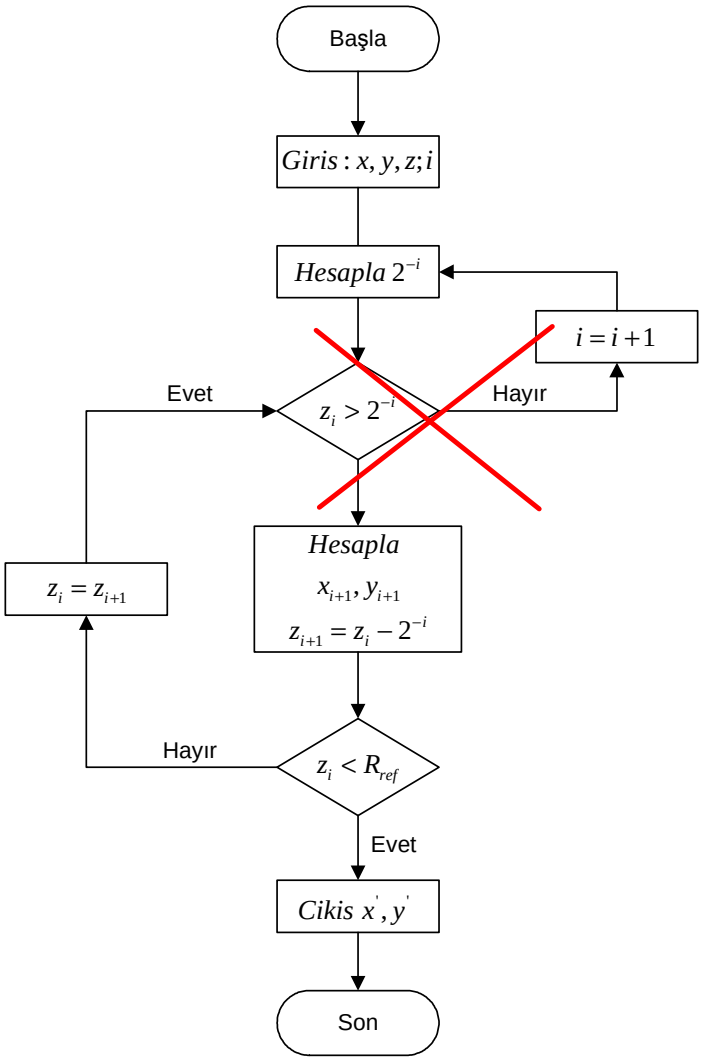
Bu durum Tablo 4-9'da gösterilmiştir.

Lojik-1 Pozisyonu	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Rotasyon adımı (i)	4,4 4,4	4,4	4	5	6	7	8	9	10	11	12	13	14
Aktif/Pasif Rotasyon adımları	**1**	0	**1**	**1**	0	0	**1**	0	**1**	0	**1**	**1**	**1**

Tablo 4 - 9. 20 derecelik açı için temel rotasyonların seçimi

Şekil 4-13'de gösterildiği gibi her bir kayan yazmaç (shift register) dizisi biti, uygun temel rotasyon adımını aktiflemek için işaretsiz 13-bitlik ϕ açısından beslenmektedir.

Kayan yazmaçlar öyle seçilmelidir ki her aktiflenmiş rotasyon adımı bir saat çevriminde yürütülmelidir. Bu mimari ile seçim algoritmasındaki z_i ve 2^{-i} ile $Rref$ karşılaştırma işlemleri ortadan kaldırılmış olur. Böylece veri yolundaki açı yakınsamasındaki tüm aritmetik işlemler ortadan kaldırılmıştır.



Şekil 4 - 15. Seçim algoritması karşılaştırma işlemlerinin iptal edilmesi

3. Çıkış Birimi:

CORDIC mimarisindeki en son birimdir. “Çıkış Birimi” mimari birimleri temsil eden Şekil 4-7’de gösterilmiştir.

Çıkış birimi; $1/\sqrt{2}$ ölçek faktörünün hesaplanmasını ve vektör ile kompanzasyonunu sağlar. Örneğin ilk quadrant için B ve C alanları içersindeki hedef açı hesaplamalarında ölçek faktörü hesaplaması bu birimde gerçekleşir.

Bu sabit ölçek faktörü basitçe ötele ve topla operasyonları ile gerçekleştirilir.

$$1/\sqrt{2} = 2^{-1/2} - 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-14} \qquad (4.27)$$

Çıkış Birimi; 2 adet $1/\sqrt{2}$ sabiti ile ölçeklenmiş birimden ve 2 adet toplama/çıkarma biriminden oluşur. Ölçekleme birimlerinin herbiri 5 adet 16-bitlik toplayıcı(adder)

biriminden oluşmaktadır. Dolayısıyla tüm donanım 12 adet 16-bitlik toplama/çıkarma biriminden oluşmaktadır.

Çıkış bloğu quad ve domain sinyallerine bağlı olarak çıkışında x ve y sinyallerini üretir. Bu sinyaller alan bilgisine göre $1/\sqrt{2}$ ile ölçeklenmiş ya da ölçeklenmemiş durumdadır. Çıkış birimindeki tüm operasyonlar bir saat çevriminde tamamlanır.

[10] algoritması için (4.23) ve (4.24) nolu denklemlerde (Alan B, Alan C) $1/\sqrt{2}$ ölçek faktörü kullanılmakta iken (4.22) ve (4.25) nolu denklmlerde (Alan A, Alan D) $1/\sqrt{2}$ ölçek faktörü kullanılmamıştır.

4.2.2.2.2 Modifiye Edilmiş Gerçekte Ölçekleme Bağımsız Adaptif CORDIC Dönüştürücü Modeli'nin Performans Analizi

4.2.2.2.2.1 Hata Analizi

Sayısal devrenin yürütülmesinde iki ana hata kaynağı vardır.

Artimetik operasyonda sınırlı kelime uzunluğu kullanıldığı için:

- Açı yakınsama hataları (Giriş kelimesi quantalama hataları)
- Yuvarlama hataları (Cut-off rounding error)

[10] için açı yakınsama hatası (angle approximation error) $2^{-16} \cong 0$ [9] ile aynı iken, Klasik CORDIC'de [1] açı yakınsama hatası 10^{-4} seviyesindedir.

Donanım uygulamasında gözlenen en baskın hata yuvarlama hatasıdır. (rounding error) Her temel CORDIC mikro rotasyonundan sonra yuvarlama hatası oluşur. Bu hata pipeline yapının her kısmında toplanarak baskın bir hata haline gelir. Sonuçta 'x' ve 'y' değerlerinin hesaplanmasında baskın bir hata oluşur.

Not: Burada sözü edilen "baskın hata" tanımı yöntemin kullanışsızlığını ifade etmez. Toplam hata değerinde oransal olarak karşılaştırdığımızda yuvarlama hatalarının açı yakınsama hatalarına göre daha yüksek değerlerde olduğu vurgulanmaktadır.

4.2.2.2.2.2 Alan Gereksinimi

Temel CORDIC pipeline biriminin 12 rotasyon adımının her biri için: 4 x16-bit adder

Quad / Alan tespit birimi için :2 adet 16-bit adder ve 2 comparator. Comparator'lerin her biri için 4-bit adder

Çıkış Birimi için: 2 adet 16-bit adder ve 2 adet ölçekleme birimi ($1/\sqrt{2}$) için

Ölçekleme birimlerinin herbiri 5 adet 16-bitlik toplayıcı(adder) biriminden oluşmaktadır.

Toplam Alan = 62 det 16-bit adder = 1000 full adder

$(12x4+2+2+5x2)$ adet 16-bit adder

Koushik'in yeni CORDIC modeli [10] ile Klasik CORDIC algoritmasını karşılaştıracak olursak;

Koushik'in CORDIC modeli [10] yaklaşık %22 daha az full adder kullanımına ihtiyaç duyar.

Koushik'in modelinde gerekli register sayısı 598'dir. Ölçek faktör içeren Klasik CORDIC algoritmasında [1], 1280 register'a ihtiyaç duyulur.

Koushik'in yeni yaklaşımı Klasik CORDIC'e göre %53 daha az register kullanımını gerektirmektedir.

Alan/Quadrant Tespit Birimi, diğer algoritmalarda kullanılan "değişken azaltım" (argument reduction) teknikleri ile karşılaştırıldığında alan gereksinimi oldukça düşüktür.

4.2.2.2.2.3 Hız

"Pipeline çalışan CORDIC İşlemcisi" biriminde; rotasyon yönünün işaretini bulmak için herhangi bir zaman harcamaya gerek yoktur. Çünkü residue açısının işareti vektör rotasyonunun yönünü belirtir. Bu özellik işlemcinin hızını önemli bir şekilde artırır. Bu sayede algoritmada her temel mikro rotasyonda rotasyon yönü için seçim operasyonuna gerek olmayacaktır.

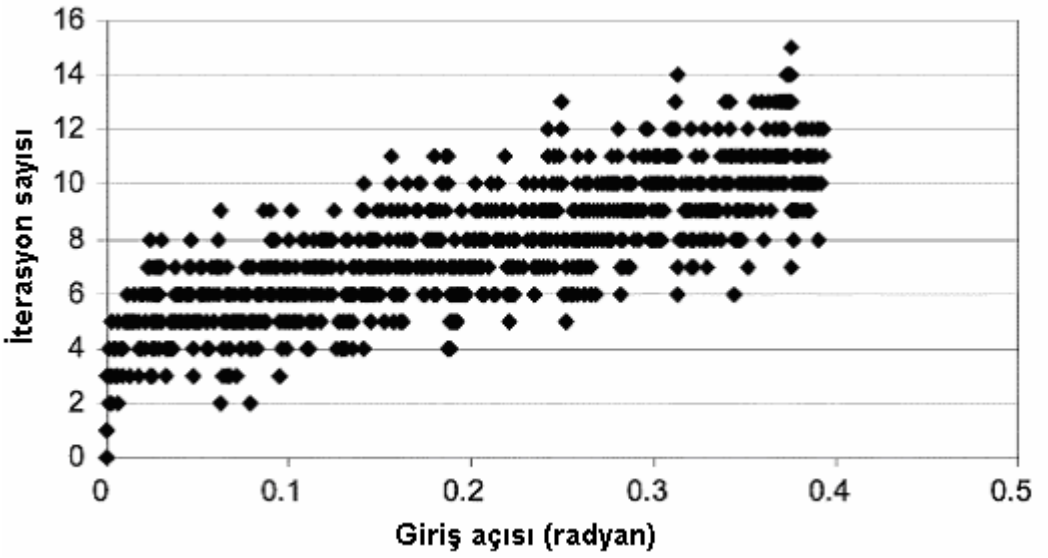
4.2.2.2.2.4 Güç

Hedef açıya (θ) yakınsama için gerekli iterasyon sayısının ve aritmetik işlem sayısının azaltılmış olması, Koushik'in yeni yaklaşımının [10], Klasik CORDIC algoritması [1] ile karşılaştırıldığında daha az güç harcamasını sağlar.

4.2.2.2.2.5 İterasyon Sayısı

Hedef açıya (θ) yakınsama için gerekli iterasyon sayısı Koushik'in yeni yaklaşımında [10], Klasik CORDIC'e [1] göre daha azdır. Koushik'in yeni modeli, n=16 bit için, klasik CORDIC'e göre yaklaşık %50 seviyelerinde daha az iterasyonda sonuca ulaşır.

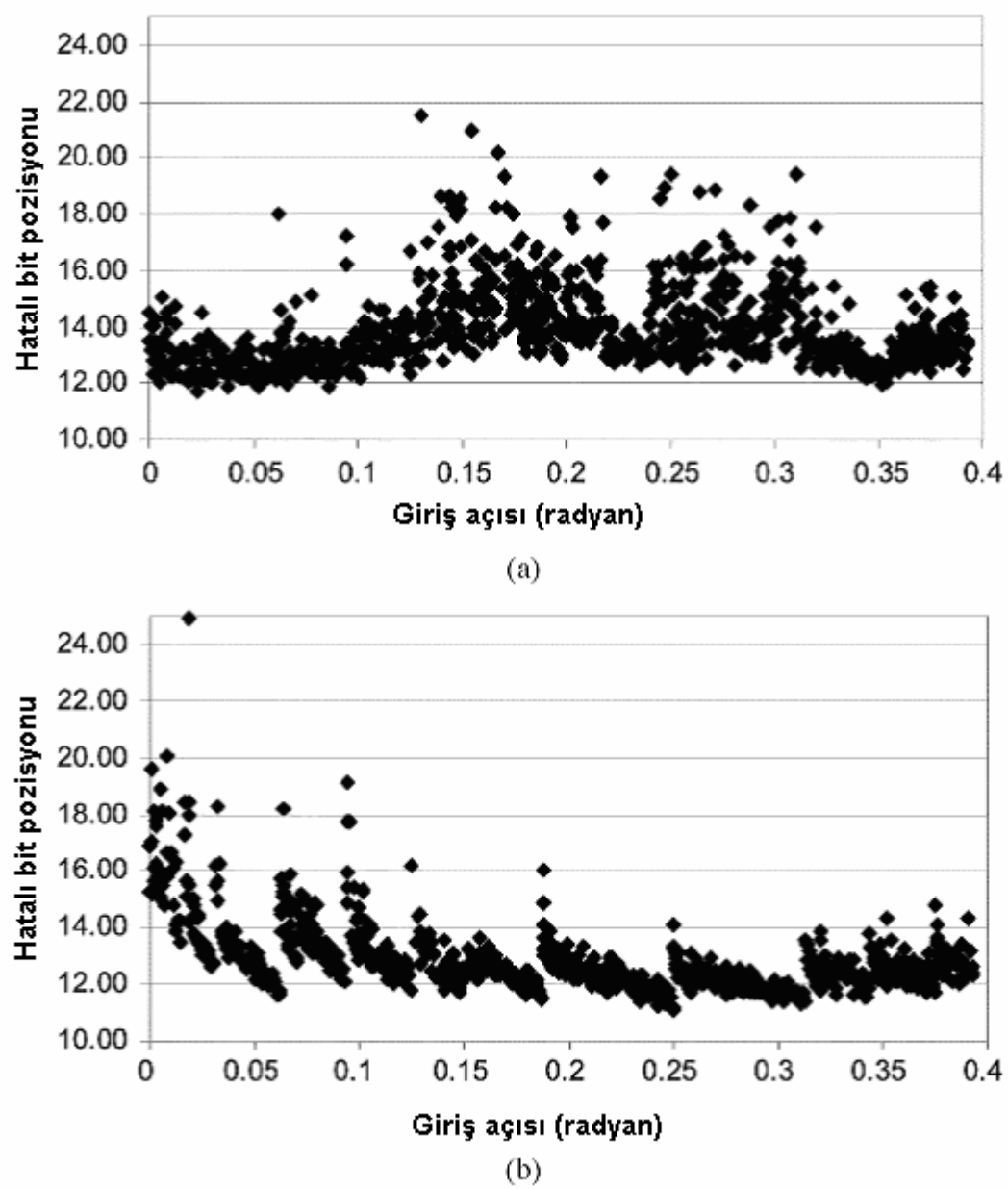
Kartezyen koordinat düzelminde ilk quadrant boyunca B ve C alanlarında kullanılan sabit K ölçek faktörü iterasyon sayısından bağımsızdır. (Klasik CORDIC'de K ölçek faktörü iterasyon sayısına bağımlı olarak elde edilmektedir.) [10]'da A ve D alanlarında ölçek faktörü kullanılmamaktadır.



Şekil 4 - 16. [10] algoritmasının yürütülmesi için ihtiyaç duyulan iterasyon sayılarının [0, 0.4) radyan değer aralığındaki dağılımı

4.2.2.2.2.6 Doğruluk Oranları

Her açı için doğruluk seviyesi Klasik CORDIC ile aynıdır. (2^{-16} hata ile)



Şekil 4 - 17. (a) x veri yolundaki sayısal hatalar. (b) y veri yolundaki sayısal hatalar

4.2.2.2.2.7 Dezavantajları

Algoritmanın en büyük problemi, en büyük açının seçiminde CORDIC mimarisinin sınırlı kelime uzunluğuna bağlı oluşudur. Kelime uzunluğu arttıkça açı değeri azalacaktır. Dolayısıyla daha fazla pipeline rotasyon adımını birleştirerek kullanmak gerekecektir. Kullanılan pipeline rotasyon adım sayısı arttıkça ilave

toplama ve öteleme operasyonlarına ihtiyaç duyulacaktır. Bu da donanım maliyetini artıracaktır.

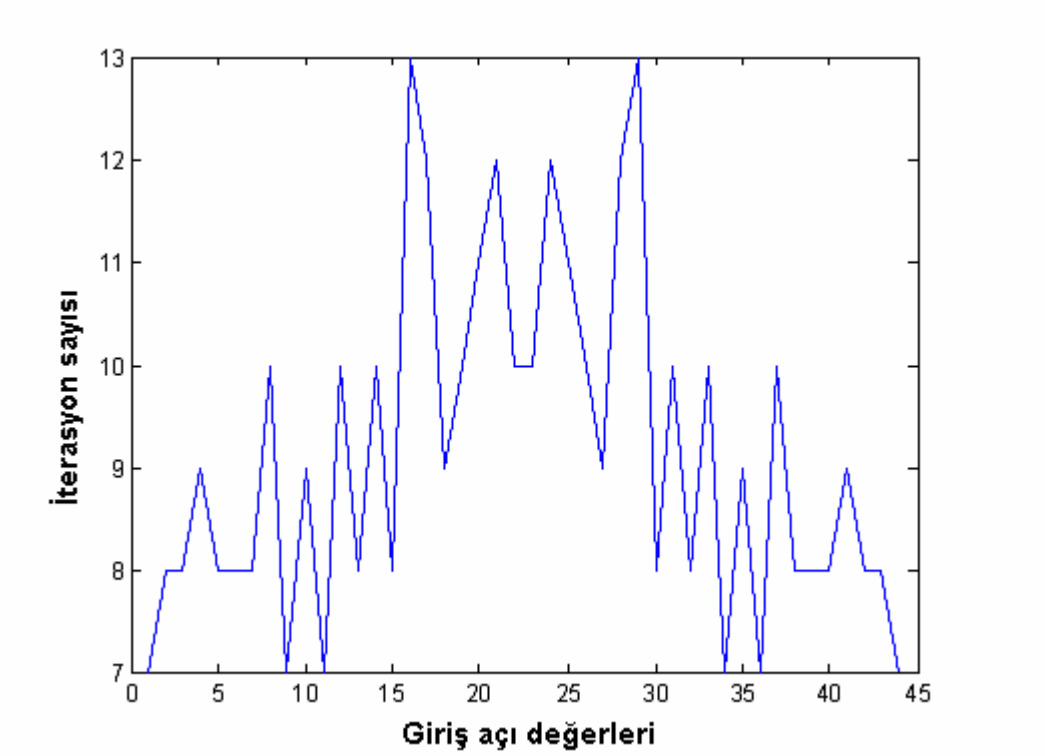
Bu teknikte 20-bit ve üzeri kelime uzunluğuna donanımın yürütülemeyeceği beklenmektedir.

4.2.2.2.2.8 Sonuç

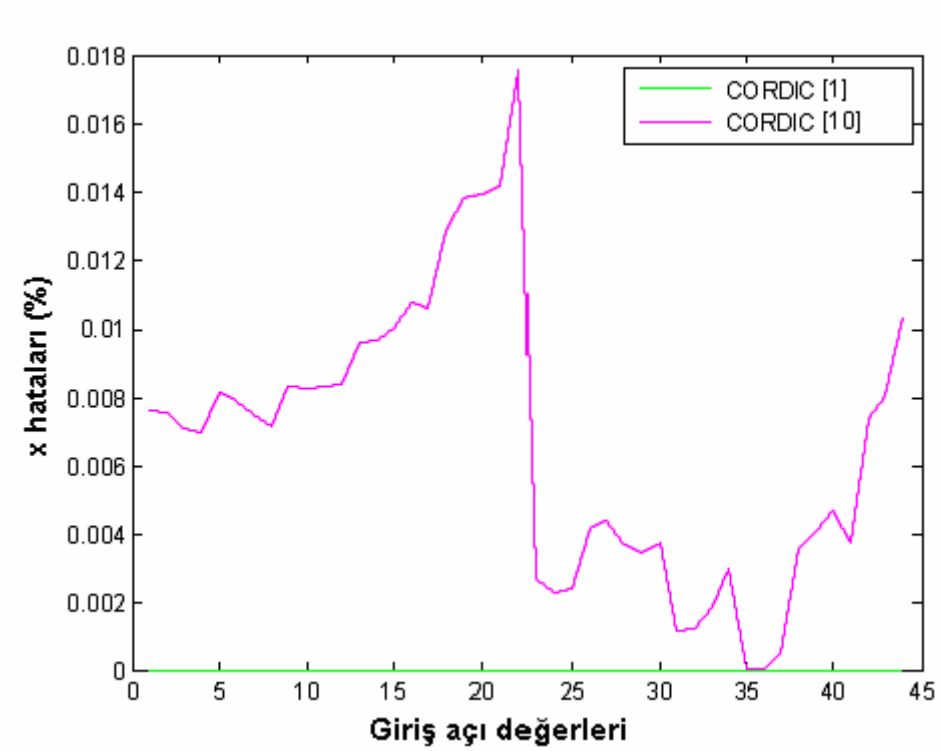
Algoritma özellikle 16-bit uygulamalar için Klasik CORDIC algoritması ile karşılaştırıldığında oldukça verimlidir. Koushik'in [10]'da önerdiği CORDIC yaklaşımı sadece dairesel vektör rotasyonları için geçerlidir.

Bu yaklaşımda [10] uygun iterasyon basamakları seçilerek hedef açıya minimum sayıda iterasyonda ve yüksek doğruluk oranlarında yakınsama gerçekleşir. Uygun iterasyon adımlarının seçimi ölçek faktörünün sonuçtaki değerini değiştirmez. Böylelikle gerçekteki ölçek faktörü iterasyon basamaklarından bağımsızdır.

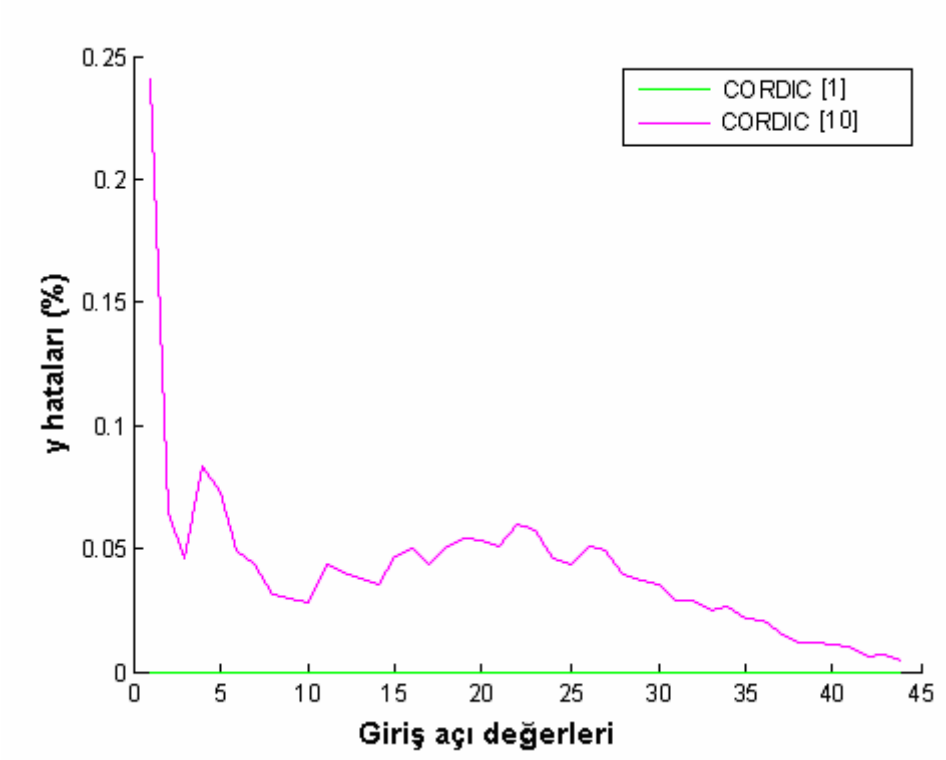
Donanımda az yer tüketilmesi ve hız artışı yönünden algoritma diğer algoritmalara göre önemli bir avantaj sağlamaktadır. "GUICORDIC" MATLAB programında n=16 için [10]'da açıklanan CORDIC yöntemini [0-45) derece açı aralığında simüle ettiğimizde gözlenen iterasyon sayısının giriş açı değerlerine göre dağılımı ile x ve y koordinat bileşenleri için hata oranları takip eden şekillerde sırasıyla verilmiştir.



Şekil 4 - 18. n=16 için (0-45) derece aralığındaki açı girişleri için ihtiyaç duyulan iterasyon sayısı dağılımı



Şekil 4 - 19. n=16 için (0-45) derece aralığındaki açı girişleri için oluşan kosinüs hatalarının dağılımı (%)



Şekil 4 - 20. n=16 için (0-45) derece aralığındaki açı girişleri için oluşan sinüs hatalarının dağılımı (%)

4.2.3. Düşük Güç Tüketen CORDIC Çekirdeği Algoritması (Zhang)

Zhang'in CORDIC modeli [11], Koushik'in [9]'da öne sürdüğü "ölçekleme bağımsız" yaklaşımını temel almıştır. Zhang, Koushik'in modelinde [9] çok küçük bir değişim gerçekleştirilerek yeni bir CORDIC modeli önermiştir. Bu küçük değişim sinüs fonksiyonunun Taylor serisi açılımındaki yakınsama değerinin farklı bir şekilde ifade edilmesi ve ölçek faktörü kullanımından kaçınılmasıdır. Önerilen CORDIC algoritmasının [11] bir diğer önemli özelliği algoritmanın 2. iterasyondan itibaren yürütülüyor olmasıdır.

Önerilen bu CORDIC yaklaşımında [11] Koushik'in modelindeki iterasyon sayısını azaltmak amaçlanmıştır.

4.2.3.1. Rotasyonel Denklemlerin Elde Edilmesi

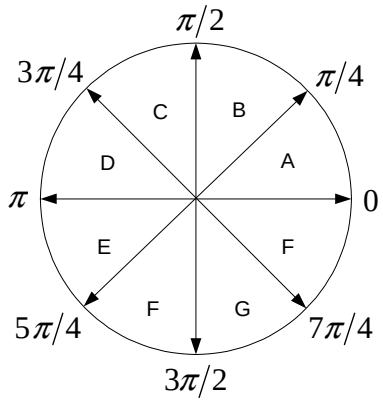
(4.2) nolu denklemde ifade edilen sinüs fonksiyonu için Taylor serisi açılımında 3. kuvvet göz önüne alınıp (1/3!) terimi 2^{-4} terimine yakınsanırsa, algoritma iterasyon sayısı bakımından [9]'a göre daha iyi bir performans sağlar. Yalnız bir iterasyon

adımı için 2^{-4} terimi Denklem 4.9'da verilen rotasyonel denklemine ilave edildiğinde [11]'de önerilen CORDIC rotasyonel denklemi elde edilir.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1-2^{-(2i+1)} & -(2^{-i}-2^{-(3i+4)}) \\ 2^{-i}-2^{-(3i+4)} & 1-2^{-(2i+1)} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.28}$$

$$z_{i+1} = z_i - r_i.2^{-i}$$

[11]'de hedef açıya yakınsamada [10]'da olduğu gibi alan katlama tekniği kullanılmakta olup, farklı quadrantlar için vektör bileşenleri, hedef açısının bulunduğu alana göre hesaplanmaktadır. Bu CORDIC tasarımında [11] koordinat düzlemi Şekil 4-21'de gösterildiği gibi 8 alana bölünmüştür.



Şekil 4 - 21. Koordinat uzayında [11] için alan katlama tekniğinin kullanımı

Zhang'in CORDIC mimarisi [11] ön işlemci (pre-processor) ve son işlemci (post-processor) birimlerinden oluşmaktadır.

Ön işlemcide θ açısının hangi alanda olduğu bilgisi tespit edilmektedir. Son işlemcide ise x ve y vektör koordinat bileşenleri hesaplanmaktadır. Bu durum Tablo 4-10'da gösterilmiştir.

	ÖN İŞLEMCİ	SON İŞLEMCİ
ALAN (DOMAIN)	θ	$[(x, y)]$
A $0 \le \phi \le \frac{\pi}{4}$	ϕ	$x' = x$ $y' = y$
B $\frac{\pi}{4} < \phi \le \frac{\pi}{2}$	$(\pi/2) - \phi$	$x' = y$ $y' = x$
C $\frac{\pi}{2} < \phi \le \frac{3\pi}{4}$	$\phi - (\pi/2)$	$x' = -y$ $y' = x$
D $\frac{3\pi}{4} < \phi \le \pi$	$\pi - \phi$	$x' = -x$ $y' = y$

Tablo 4 - 10. Hedef açısının tespiti ve vektör bileşenlerinin hesaplanması

Zhang'in CORDIC modelinde [11], [9]'da açıklanan seçim algoritması mantığı kullanılmakta olup, maksimum iterasyon sayısı n değişken değerine bağlıdır. Tablo 4-11'de n=10 iterasyon için quantalanmamış ve quantalanmış mikro rotasyon açı değerlerinin iterasyon sayısına göre almış olduğu değerler ve quantalama hata oranları verilmiştir.

i	Quantalanmamış mikro rotasyon açısı $\alpha_i = \sin^{-1}(2^{-i})$ (Radian)	Quantalanmış mikro rotasyon açısı $\alpha_i = 2^{-i}$ (Radian)	Quantalama Hatası $(\%)$
0	1.570796	1	36.3380
1	0.523598	0.5	4.50726
2	0.252680	0.25	1.06063
3	0.125328	0.125	0.26171
4	0.062541	0.0625	0.06556
5	0.031255	0.03125	0.01600
6	0.015626	0.015625	0.00640
7	0.007812	0.0078125	0.00128
8	0.003906	0.00390625	0.00026
9	0.001953	0.00097656	0

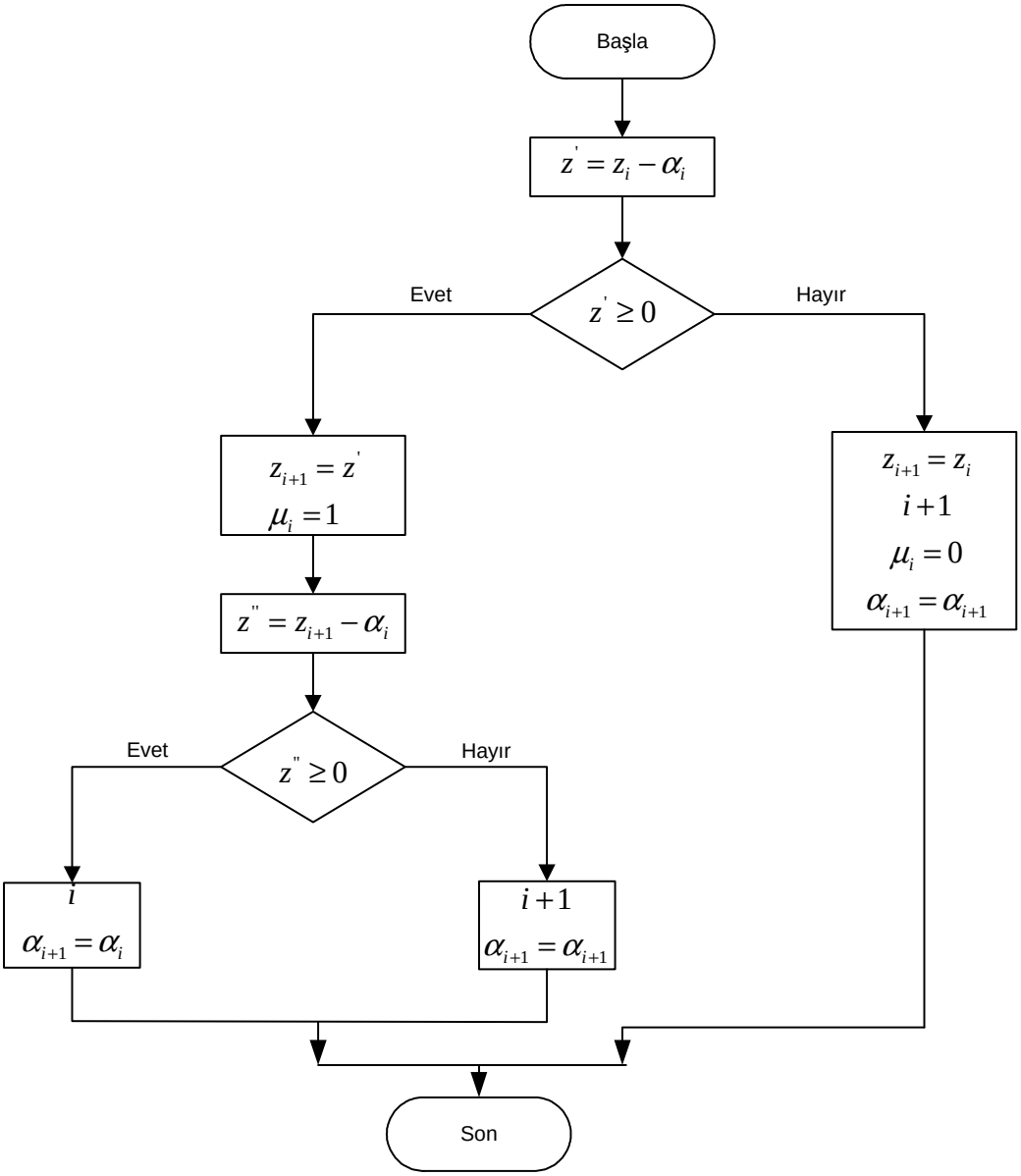
Tablo 4 - 11. Mikro rotasyon açıları için quantalama hataları

Tablo 4-11 incelendiğinde α_0 ve α_1 başlangıç mikro rotasyon açıları için sırasıyla %36.3 ve %4.5 quantalama hatalarının mevcut olduğu görülmektedir. Algoritmanın α_0 (sıfırıncı) veya α_1 (birinci) mikro rotasyon açısı ile yürütülmeye başlaması

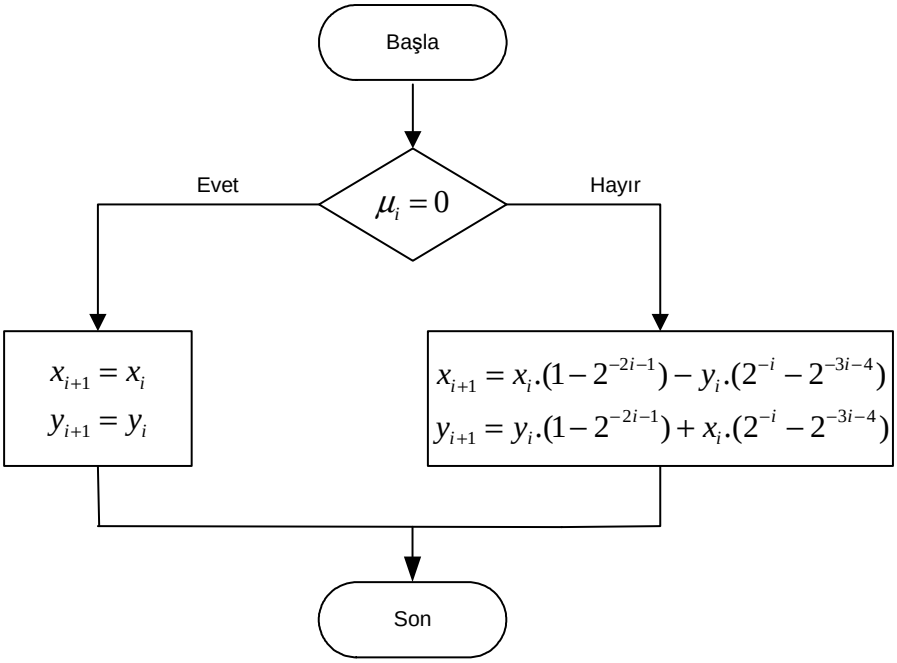
durumunda yüksek quantalama hataları meydana gelecek ve yanlış vektör sonuçları üretilecektir. Takip eden mikro rotasyon açıları için hata değerleri azalmaktadır. Yüksek doğruluk istenen tasarımlarda CORDIC rotasyonları büyük mikro rotasyon açıları ile rotasyona başlamamalıdır. Bu sebeple Zhang öne sürdüğü modelde [11] algoritmasını α_2 başlangıç mikro rotasyon açısı ile yürütülecek şekilde düzenlemiştir.

Zhang'in değerlendirmesi: Birinci iterasyonda ki % 4.5.lik quantalama hatası aşırı hassaslık gerektirmeyen uygulamalarda, özellikle donanımda kullanılan alanı azaltmak ve algoritma hızını artırmak istediğimiz zamanlarda hesaba katılabilir. Ancak yüksek quantalama hatası oluşumundan dolayı tercih edilmemektedir. Bu sebeple önerilen algoritma ikinci iterasyondan (i=2) itibaren yürütülmekte olup, yüksek quantalama hatalarına sahip ilk iki iterasyon (i=0 ve i=1) hesaba katılmamaktadır.

Zhang'in CORDIC modelinde seçim algoritmasına göre herhangi bir bit birden fazla tekrarlanabilir. Şekil 4-22'de algoritmanın [11], hedef açıya yakınsama mantığını gösteren akış diyagramı verilmişken, x ve y vektör koordinat değerlerinin hesaplanmasına ilişkin akış diyagramı Şekil 4-23'de verilmiştir.



Şekil 4 - 22. Hedef açıya yakınsama operasyonları



Şekil 4 - 23. x ve y rotasyonlarına ait akış diyagramı

30^0'lik rotasyon açısı için CORDIC operasyonlarının nasıl gerçeklendiği ÖRNEK-4'de açıklanmıştır.

ÖRNEK-4:

GİRİŞ VERİSİ: $(x, y)^T$ ve rotasyon açısı $\theta = 30^O$ açısı

Herhangi bir (x, y) koorinat bileşenlerine sahip bir vektörü $\theta = 30^0$ rotasyon açısı kadar döndürdüğümüzde elde edeceğimiz $(x', y') = (\cos\theta, \sin\theta)$ ifadesi aşağıda açıklandığı gibi hesaplanmaktadır.

30 derecelik bir hedef açısını hesaplayabilmek için Klasik CORDIC algoritmasında [1], 10 iterasyona ihtiyaç vardır. Bu durum ÖRNEK-1'de ifade edilmiştir. 30 derecelik bir hedef açısını hesaplayabilmek için Zhang'in CORDIC modelinde [11], 4 iterasyonda sonuca ulaşılmaktadır. Zhang'in CORDIC algoritması kelime uzunluğunu 32 bit'e kadar kullanılabilecek yetenektedir.

Koushik'in yaklaşımında [10], MSB bitler olan 10. , 11. ve 12. bitlerin dışındaki 0-9 arasındaki 10 adet bit yalnız bir defa kullanılmakta iken Zhang'in CORDIC modelinde seçim algoritmasına göre herhangi bir bit birden fazla tekrarlanabilir.

Zhang'in modelinde $\theta = 30^0$ açısına ulaşabilmek için başlangıç iterasyonu olan 2. iterasyon iki kez yürütülmüştür. 6. ve 7. iterasyon adımları ise birer kez yürütülmüştür. Tablo 4-12 ve 4-13'de iterasyon adımlarının seçimi gösterilmiştir.

i	$\alpha_i = \sin^{-i}(2^{-i})$ **(Derece)**	**Yürütülen iterasyon adımı**	**Yürütülen iterasyon sayısı**
2	14.32	EVET	2
3	7.162		
4	3.581		
5	1.790		
6	0.895	EVET	1
7	0.447	EVET	1
8	0.223		
9	0.112		
10	0.056		

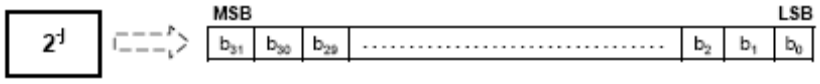
Tablo 4 - 12. 30 derecelik hedef açısı için yürütülen mikro rotasyon adımları

	i	$z_i - \alpha_i = z_{i+1} \quad (Degrees)$	
Başlangıç Iterasyonu: 2	2	+30 - 14.32 = +15.68	Toplam İterasyon Sayısı: 4
	2	+15.68 - 14.32 = +1.36	
	6	+1.36 - 0.895 = +0.47	
	7	+0.47 - 0.447 = +0.023	

Tablo 4 - 13. 30 derecelik hedef açısı için yürütülen CORDIC operasyonları

ÇIKIŞ VERİSİ: $\left[x^{'}, y^{'}\right]^T \quad x^{'} = x.\cos 30^0 - y.\sin 30^0 \;,\; y^{'} = y.\cos 30^0 + x.\sin 30^0$

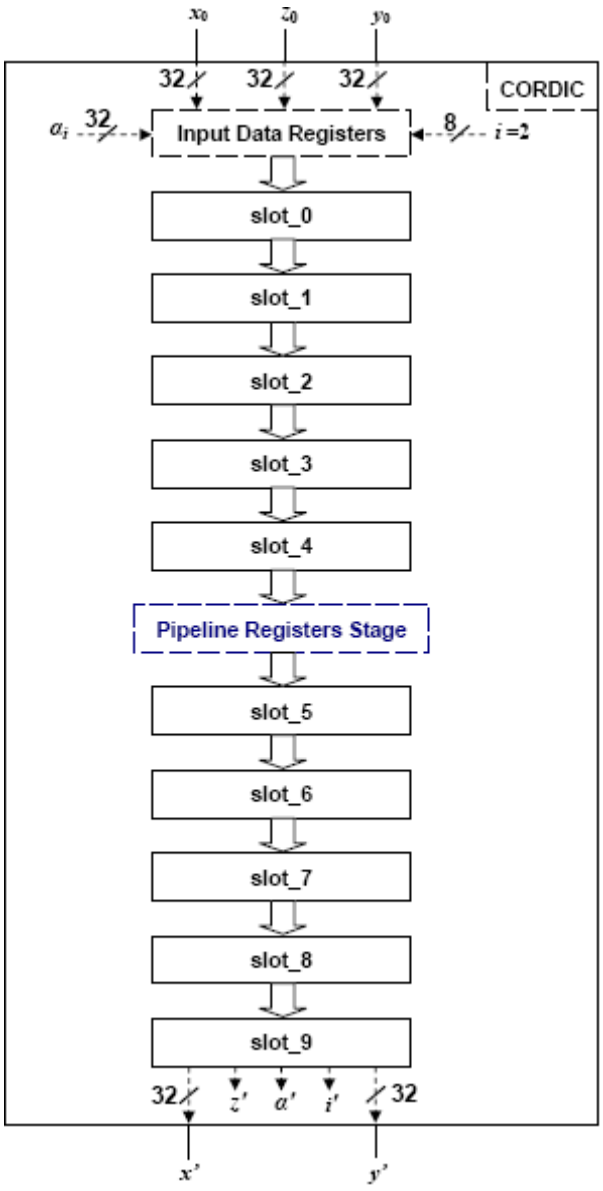
$i \in \{0, 1, 2, \cdots, 9\}$ iken $\mu_i = [1, 1, 0, 0, 0, 1, 1, 0, 0, 0]$



Şekil 4 - 24. 32-bit kelime uzunluğu işlemler için temel öteleme yapısı görünümü

[11] için CORDIC IP çekirdeğinin orijinal mimarisi 10 slot bloğu ve bir pipeline rotasyon adımından oluşmaktadır. Bu mimari yapı Şekil 4-25'de gösterilmiştir.

IP çekirdeği girişleri register'da tutulur. Bu tasarımda çıkış register'ları bulunmamaktadır. Tüm register'lar global bir saat kaynağı ile sürülmekte olup tümü senkron reset'e sahiptir.



Şekil 4 - 25. CORDIC IP Core dahili yapısı

x' ve y' sırasıyla çıkış değişkenleridir. Tasarımdaki pipeline register rotasyon adımlarından dolayı çıkış verisi, giriş verisine göre bir saat çevrimi kadarlık gecikmeye sahiptir.

Tasarımda toplam 10 adet slot bloğu kullanılmakta olup maksimum rotasyon sayısı 10 değerine ayarlanmıştır. Bu çekirdek MATLAB simülasyonu ile aynı doğrulukta gerçeklenmiştir. (sabit noktalı gösterimin yuvarlama hataları ihmal

edildiğinde) Tüm veri sabit noktalı (fixed-point) 2'nin tersi formatda temsil edilmiştir.

Giriş verisi, çıkış verisi ve dahili değişkenlerin kelime uzunlukları 32-bit tasarlanmışken, 'i' değişkeni 8-bit tasarlanmıştır. Çünkü maksimum rotasyon sayısı 10 değerine ayarlanmıştır. 'i' değişkeni 2. mikro rotasyondan başlatıldığında 'i' değişkeninin maksimum değeri 12 değerini aşamayacaktır.

ÖRNEK-5:

Tüm girişler onluk sayı sisteminden sabit noktalı ikili sayı sistemine dönüştürülmüş olarak MATLAB'da temsil edilmektedir. 32-bit tasarım için en anlamlı MSB işaret biti olarak kullanılmakta iken diğer 32-bit onluk sayının ikili karşılığını her rotasyon açısında temsil etmektedir. x' ve y' bileşenlerinin çıkış verisini kontrol etmek için ilk olarak θ hedef açısı ikili sayı sisteminden onluk sayı sistemine dönüştürülür ve 2^p terimine bölünür. p onluk sayı sisteminde temsil edilen değerde kullanılan bit sayısını vermektedir. Sırasıyla x' ve y' çıkış verilerinin tahmini için $\cos(\theta)$ ve $\sin(\theta)$ değerleri hesaplanabilir. Sonuçta x' ve y' çıkış verileri aynı zamanda $\cos(\theta)$ ve $\sin(\theta)$ değerlerine karşılık gelecektir.

$$(\theta)_2 = (00000010001110111110100011010100)_2 \quad \text{radians}$$

$$(\boldsymbol{\theta})_{\mathbf{10}} = (37480660)_{10} \bullet\bullet 2^{\mathrm{P}} \quad \text{radians}$$

$$= (37480660)_{10} \bullet\bullet 2^{31} = \mathbf{0.0175} \text{ radians}$$

$$\mathbf{cos}\ \boldsymbol{\theta} = \cos(0.0175) = \mathbf{0.9998}$$

$$(x')_2 = (01111111111110110011011100000100)_2$$

$$(\boldsymbol{x'})_{\mathbf{10}} = (2147200000)_{10} \bullet\bullet 2^{\mathrm{P}} = (2147200000)_{10} \bullet\bullet 2^{31} = \mathbf{0.9999} \approx \mathbf{cos}\ \boldsymbol{\theta}$$

$$\mathbf{sin}\ \boldsymbol{\theta} = \sin(0.0175) = \mathbf{0.0175}$$

$$(y')_2 = (00000010001011111111110001011100)_2$$

$$(\boldsymbol{y'})_{\mathbf{10}} = (36699228)_{10} \bullet\bullet 2^{\mathrm{P}} = (36699228)_{10} \bullet\bullet 2^{31} = \mathbf{0.0171} \approx \mathbf{sin}\ \boldsymbol{\theta}$$

Şekil 4 - 26. Kosinüs ve sinus değerlerinin hesaplanması

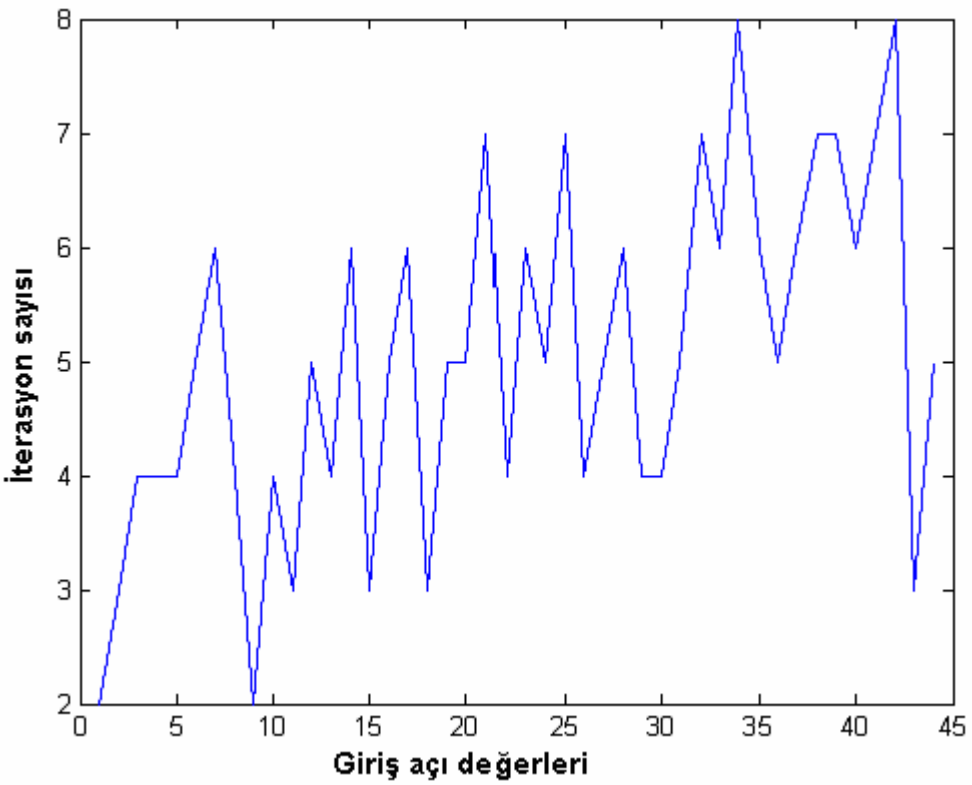
Yukarıdaki örnekte verilen rotasyon açısı için x' ve y' çıkış verilerinin hesaplanışı 32-bit mimari için görselleştirilmiştir. Zhang'in CORDIC mimarisinin avantajları; Koushik'in mimari yapısı yerine klasik seçim algoritmasının kullanılması, [10] ile karşılatırıldığında 16 bit'den 32 bit'e algoritmanın kelime uzunluğunu artırabilecek bir imkanı sağlamıştır ki, Zhang'de yeni modeli 32 bite göre tasarlanmıştır. [10]'da açıklanan CORDIC mimarisi 20-bit kelime uzunluğu ile sınırlıdır. Ayrıca [11], [10] ile karşılaştırıldığında daha düşük iterasyon sayısında sonuca ulaşabilmektedir.

4.2.3.2. Dezavantajları

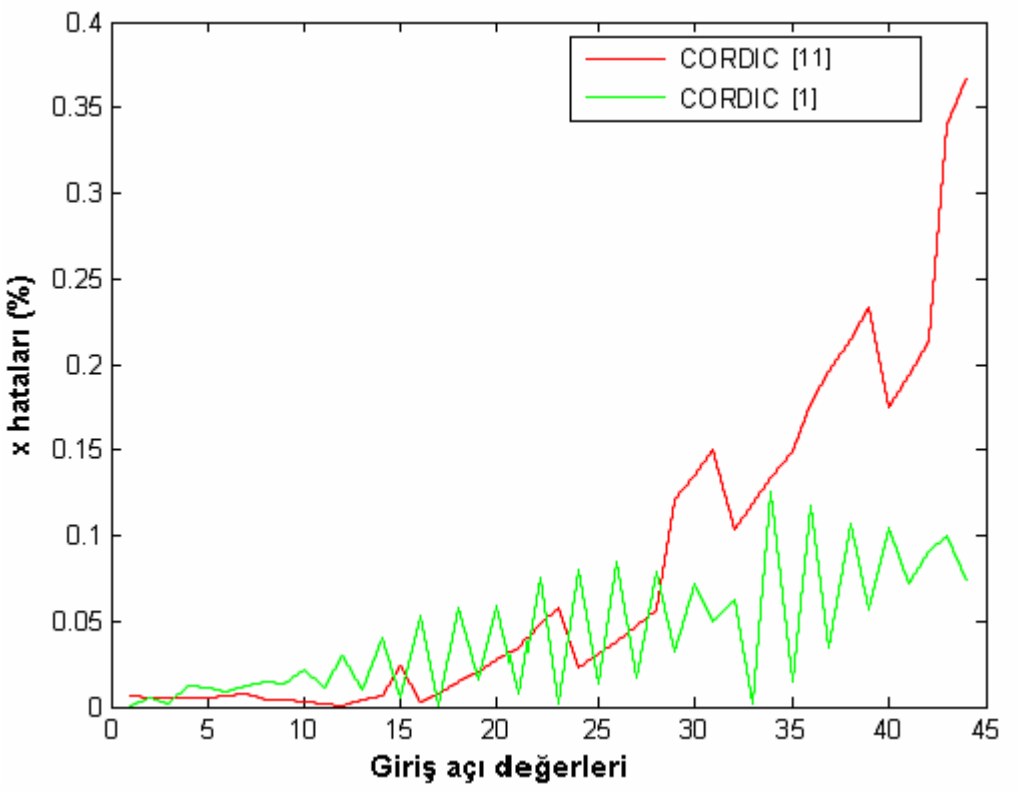
Koushik'in öne sürdüğü yeni yaklaşımdaki [10] mimari yapı yerine, [9]'da öne sürülen klasik seçim algoritmasının kullanılması algoritmanın doğruluk ve hız karakteristikleri açısından en uygun referans değerine göre yürütülmesi zorunluluğunu ortaya çıkarmaktadır.

4.2.3.3. Sonuç

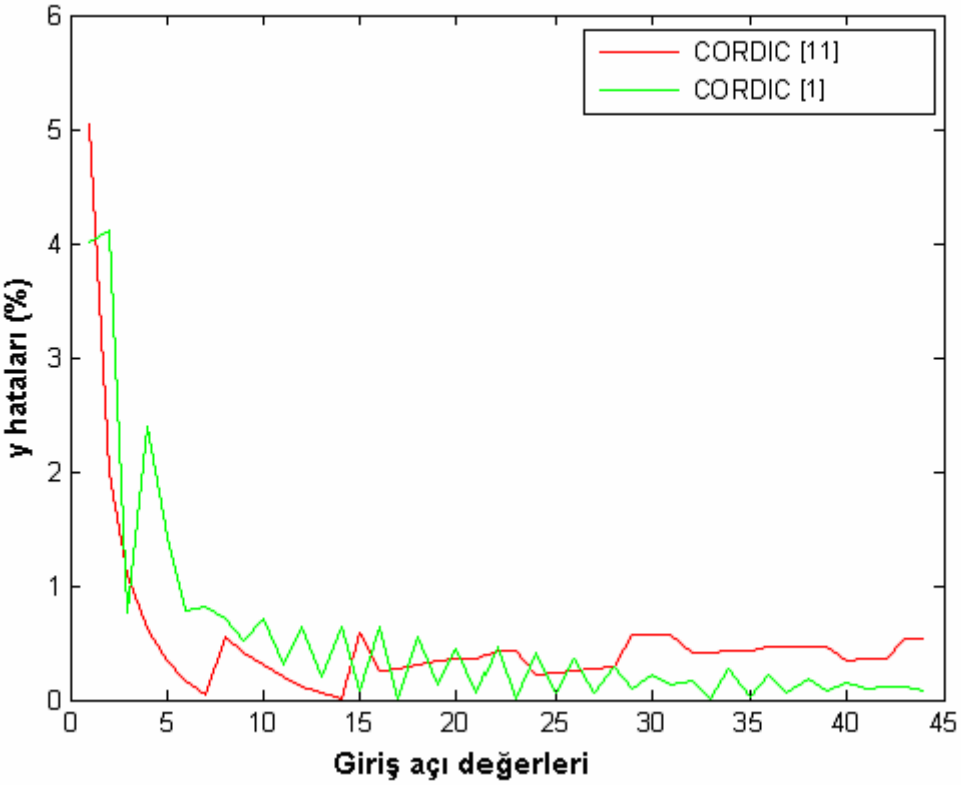
[11] için maksimum iterasyon sayısı 10 olarak seçilip, algoritma (0-45) derece aralığındaki açı değerleri boyunca "GUICORDIC" MATLAB programında simüle edilirse, ihtiyaç duyulan iterasyon sayısının ilk quadrant boyunca dağılımı ve oluşan kosinüs-sinüs hataları takip eden şekillerde gösterilmiştir.



Şekil 4 - 27. n=10 için (0-45) derece aralığındaki açı girişleri için iterasyon sayılarının dağılımı



Şekil 4 - 28. n=10 için (0-45) derece aralığındaki açı girişleri için oluşan kosinüs hatalarının dağılımı (%)



Şekil 4 - 29. n=10 için (0-45) derece aralığındaki açı girişleri için oluşan sinüs hatalarının dağılımı (%)

4.2.4. Yüksek Doğruluklu Azaltılmış İterasyonlu CORDIC İşlemci Algoritması

[12]'de açıklanan CORDIC yöntemi bu tez çalışmasının bir ürünüdür.

4.2.4.1. Yüksek Doğruluklu Azaltılmış İterasyonlu CORDIC İşlemci Algoritmasının Rotasyonel Denklemlerinin Gelişimi

Farklı yaklaşımları esas alan çok sayıda CORDIC algoritması Klasik CORDIC algoritmasından [1] türetilmekte olup, türetilen CORDIC yöntemlerinden biri olan [12]'de atıfta bulunulan CORDIC algoritması bu bölümde açıklanacaktır. "Yüksek Doğrulukta Azaltılmış İterasyonlu CORDIC İşlemcisi" algoritması [12], bu tez çalışmasının bir ürünüdür.

Önerilen CORDIC yönteminde [12], birinci mikro rotaston adımında düşük quantalama hatası oluşumu amaçlanmıştır. Bu yolla algoritmada büyük mikro rotasyon açılarının kullanımından doğan yüksek quantalama hataları ihmal edilebilecek kadar düşük seviyelere getirilebilir. FFT vb. yüksek doğruluk ve hızda

sonuca ulaşılmak istenen uygulamalarda [12] kullanımı avantaj sağlamaktadır. İleriki bölümlerde yukarıda amaçlanan noktaya ulaşılabilmesi için gerekli modifikasyonlar açıklanacaktır.

4.2.4.1.1 Yüksek Doğruluklu Azaltılmış İterasyonlu CORDIC İşlemci Algoritması Modifikasyonları

4.2.4.1.1.1 Mikro Rotasyon Açısı İçin Modifikasyon

[11]'de açıklanan Denklem (4.28) mikro rotasyon açılarının hesaplanması sürecinde iyileştirilebilir.
İlave toplama/öteleme operasyonları:

$$2^{-6} + 2^{-7} + 2^{-13} = 0.023559 \tag{4.29}$$

İlk iterasyon adımında daha düşük quantalama hatalarının oluşumu için Denklem (4.29) algoritmaya uygulanabilir. Bu durumda ilk iterasyondaki quantalama hata oranı %0.0074 değerine düşer.
Mikro rotasyon açısı Denklem (4.29)'de ifade edilen ilave operasyonlar ile Denklem (4.30)'daki gibi yeniden ifade edilebilir.

Modifiye edilmiş mikro rotasyon açısı:

$$2^{-1} + 2^{-6} + 2^{-7} + 2^{-13} = 0.523559 \tag{4.30}$$

Yukarıda açıklanan modifikasyonlar algoritmaya uygulanmadan önceki quantalama hata değerleri ilk üç mikro rotasyon adımı için Tablo 4-14'de verilmiştir.

İterasyonlar	**Çevrim tablosu değerleri**	**Çevrim tablosu kullanılmaksızın mikro rotasyon açı değerleri**	**Quantalama hataları**
i	$\alpha_i = \sin^{-1}(2^{-i})$ (radian)	$\alpha_i = 2^{-i}$	%
0	1.570796	1	36.338
1	0.523598	0.5	4.5068
2	0.252680	0.25	1.0607

Tablo 4 - 14. Yüksek quantalama hatalarına sahip başlangıç mikro rotasyon açıları

Tablo 4-14, Denklem (4.28) uygulaması sonucunda elde edilmiş olup, başlangıç mikro rotasyonlarındaki yüksek quantalama hataları Tablo 4-14'de gösterilmiştir.

Tablo incelendiği taktirde i=1 adımı için α_i mikro rotasyon açısı çevrim tablosu kullanımında 0.523598 değerini almakta iken, çevrim tablosu kullanılmadığı durumda 0.5 değerini aldığı gözlemlenebilir. İlk mikro rotasyon adımında oluşan %4.5 'luk quantalama hatası çok ciddi bir hata oranıdır ki algoritmanın bu adımının [9], [10] ve [11]'de tanıtılan algoritmalar için işletilmesi durumunda sonuç değerlerinde ciddi hatalar oluşacaktır. Bu nedenle ilgili algoritmalar i=1 mikro rotasyon adımını algoritmada kullanamamaktadır.

Mikro rotasyon açısı için modifikasyonlar algoritmaya uygulandıktan sonraki quantalama hata değerleri ilk üç mikro rotasyon adımı için Tablo 4-15'de verilmiştir.

İterasyonlar	**Çevrim tablosu değerleri**	**Çevrim tablosu kullanılmaksızın mikro rotasyon açı değerleri**	**Quantalama hataları**
i	$\alpha_i = \sin^{-1}(2^{-i})$ (radian)	$\alpha_i = 2^{-i}$	%
0	1.570796	1	36.338
1	0.523598	0.523559	0.0074
2	0.252680	0.25	1.0607

Tablo 4 - 15. Azaltılmış quantalama hatalarına sahip başlangıç mikro rotasyon açıları

Denklem (4.29)'deki ilave operasyonlar modiyiye edilmiş CORDIC algoritmasında [12] kullanılmakta olup hesaplamalardaki ilk iterasyon adımı göz önüne alındığında, birinci iterasyondaki yüksek quantalama hatasının kabul edilebilir bir seviyeye düşürüldüğü gözlenebilir. Tablo 4-14 ve Tablo 4-15 incelendiğinde i=1 iterasyonunda quantalama hata oranı % 4.5068'den % 0.0074 değerine düşürüldüğü anlaşılmaktadir ki bu ciddi bir iyileştirmedir.

4.2.4.1.1.2 Koordinat Bileşen Değerleri İçin Modifikasyon

Mikro rotasyon açısı için önerilen modifikasyonlar algoritmanın başarılı bir şekilde yürütülebilmesi için yeterli değildir. Ayrıca vektör koordinat bileşenleri üzerinde bir modifikasyona ihtiyaç duyulmaktadır.

Denklem (4.10)'da ifade edilen CORDIC rotasyonel deklemi, önerilen CORDIC algoritması için bir başlangıç noktası olarak kullanılabilir. (4.10) nolu "Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC" rotasyonel ve (2.19) nolu Klasik CORDIC rotasyonel denklemi "GUICORDIC" MATLAB programında kartezyen koordinat sisteminin ilk quadrantında [30-44] derece giriş açıları için simüle edildiğinde hesaplanan kosinüs, sünüs bileşen değerleri ve hata oranları Tablo 4-16'da verilmiştir.

	[9] nolu denklem		**[1] nolu denklem**		**Fark değerleri**	
Açı (θ)	Kosinüs	Sinus	Kosinüs	Sinus	Kosinüs bileşeni için	Sinüs bileşeni için
30	0.874985	0.500027	0.866085	0.500051	**0.008900**	0.000023
31	0.866123	0.515224	0.857226	0.515091	**0.008897**	0.000133
32	0.856996	0.530268	0.848105	0.529975	**0.008891**	0.000293
33	0.847609	0.545147	0.838728	0.544693	**0.008881**	0.000454
34	0.837954	0.559876	0.829092	0.559250	**0.008862**	0.000626
35	0.828054	0.574418	0.819206	0.573635	**0.008848**	0.000783
36	0.817898	0.588788	0.809068	0.587847	**0.008830**	0.000941
37	0.807496	0.602975	0.798694	0.601866	**0.008802**	0.001109
38	0.796699	0.617217	0.788068	0.615713	**0.008631**	0.001504
39	0.785811	0.631022	0.777201	0.629375	**0.008610**	0.001647
40	0.774676	0.644643	0.766108	0.642832	**0.008568**	0.001811
41	0.763284	0.658094	0.754771	0.656107	**0.008513**	0.001987
42	0.751680	0.671318	0.743205	0.669180	**0.008475**	0.002138
43	0.739845	0.684340	0.731411	0.682050	**0.008434**	0.002290
44	0.727786	0.697150	0.719407	0.694701	**0.008379**	0.002449

Tablo 4 - 16. Denklem (2.19) and (4.10) n=16 bit için yürütüldüğünde aralarında oluşan fark

Tablo 4-16 incelendiğinde tüm kosinüs bileşenleri için yaklaşık 0.0088 değerinde sabit bir hata farkının olduğu gözlemlenmiştir. Sinüs bileşenleri için gözlenen hata değerleri ise operasyon sonucuna olan etkilerininin az olması sebebiyle ihmal edilmiştir. Önerilen CORDIC rotasyonel Denklemi, Denklem (4.10)'da kosinüs bileşenlerinden 0.0088 değeri çıkartılarak son halini alır.

Öteleme ve toplama operasyonları cinsinden 0.0088 hata farkına en yakın ifade

Denklem (4.31)'de ifade edilen ($2^{-7} + 2^{-10}$) operasyonudur.

$$(0.0088 \cong 2^{-7} + 2^{-10}) \tag{4.31}$$

($2^{-7} + 2^{-10}$) öteleme ve toplama operasyonları algoritmada kolayca gerçeklenebilir.

Mikro rotasyon açı değerleri ve vektör bileşen değerleri üzerinde yapılan bu iyileştirmeler sonrasında önerilen CORDIC algoritması [12] elde edilir.

Önerilen CORDIC algoritması rotasyonel denklemleri sırasıyla (4.32) ve (4.33) nolu denklemlerde tanımlanmıştır.

İlk mikro rotasyon adımı (i=1) için önerilen CORDIC algoritması [12] rotasyonel denklemi Denklem (4.32)'de verilmiştir.

$$\begin{bmatrix} x'_{i+1} \\ y'_{i+1} \end{bmatrix} = \begin{bmatrix} 1-2^{-(2i+1)} & -2^{-i} \\ 2^{-i} & 1-2^{-(2i+1)} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x'_{i+1} - (2^{-7} + 2^{-10}) \\ y'_{i+1} \end{bmatrix}, \ (i=1) \tag{4.32}$$

$$z_{i+1} = z_i - r_i.(2^{-i} + 2^{-6} + 2^{-7} + 2^{-13}), \ (i=1)$$

Takip eden mikro rotasyon adımları (i>1) için önerilen CORDIC algoritması [12] rotasyonel denklemi Denklem (4.33)'de verilmiştir.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1-2^{-(2i+1)} & -(2^{-i} - 2^{-(3i+4)}) \\ 2^{-i} - 2^{-(3i+4)} & 1-2^{-(2i+1)} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.33}$$

$$z_{i+1} = z_i - r_i.2^{-i} \quad , \ (i=2,.....)$$

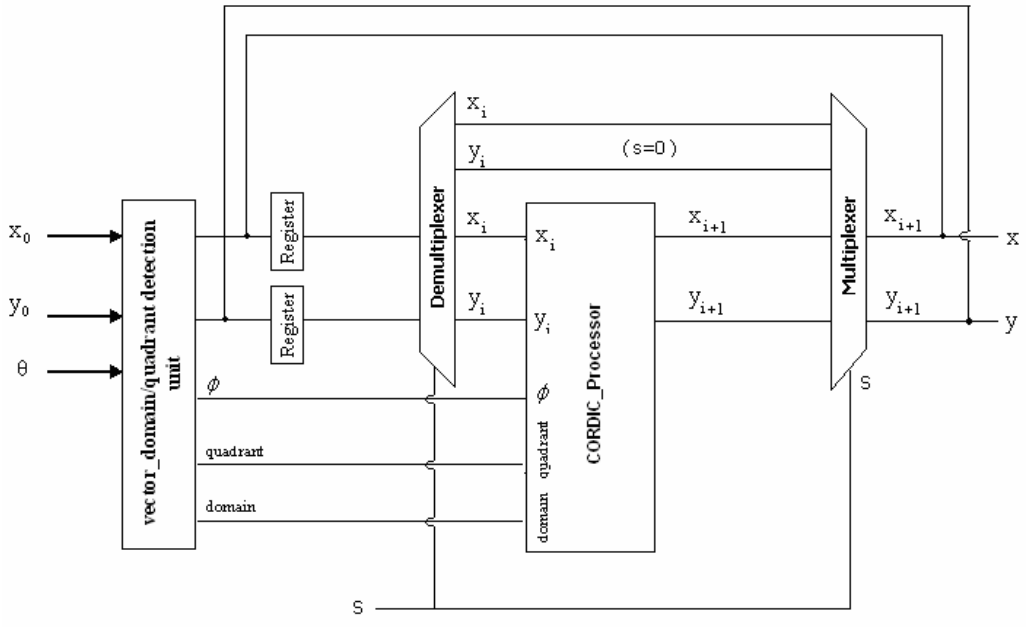
(4.32) ve (4.33) nolu CORDIC rotasyonel denklemleri, sırasıyla [10] ve [11]'de açıklanan (4.10) ve (4.28) nolu CORDIC rotasyonel denklemlerden farklıdır.

Zhang'in CORDIC algoritması [11], ikinci iterasyondan itibaren yürütülürken, Koushik'in yeni CORDIC yaklaşımı [10], dördüncü iterasyondan itibaren yürütülmektedir.

Öte yandan önerilen CORDIC algoritmasında ise algoritma birinci iterasyondan itibaren yürütülmektedir. Yukarıda açıklanan modifikasyonlar, algoritmanın ilk iterasyonda büyük mikro rotasyon açı değerleri ile yürütülebilmesini sağlamaktadır. Önerilen yöntem ile, başlangıç mikro rotasyon açısındaki (α_1) quantalama hatası ciddi oranda düşürülmüş olup, yüksek doğrulukta sonuçlara düşük sayıda iterasyon ile ulaşılmış olur. Önerilen algoritma [9], [10] ve [11]'de açıklanan CORDIC yöntemlerine göre iterasyon sayısı-doğruluk ilikisi göz önüne alındığında daha iyi bir performans sağlamaktadır.

4.2.4.1.2 Önerilen CORDIC Algoritması İçin Donanım Mimarisi

Önerilen CORDIC algoritması donanımı "vektör alan/quadrant tespit" ve "CORDIC processer" birimlerinden oluşmakta olup, CORDIC mimarisi Şekil 4-30'da gösterilmiştir. θ rotasyon açısı kadarlık vektör rotasyonunda, yeni vektör koordinat değerlerinin hesaplanabilmesi için başlangıç vektör koordinatları ve rotasyon açısı "alan /quadrant tespit" biriminin girişine uygulanır. θ hedef açısının bulunduğu quadrant ve alan bu birimde hesaplanmaktadır.



Şekil 4 - 30. Önerilen CORDIC mimarisinin basitleştirilmiş yapısı

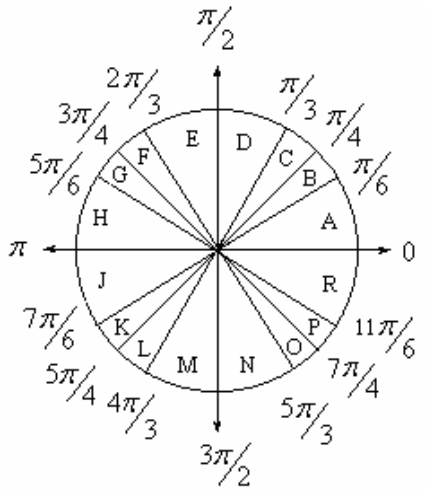
Vektör rotasyonu yürütülürse, s rotasyon aktifleme biti ‘1’ değerine ayarlanır ve (x,y) değerleri çoğullama çözücünün (demultiplexer) çıkışında bulunan CORDIC işlemciye gönderilir.

Her iterasyonda (4.32) ve (4.33) nolu denklemler kullanılarak yeni (x,y) değerleri hesaplanır. Eğer vektör rotasyonu yürütülmez ise, s rotasyon aktifleme biti ‘0’ değerine ayarlanır ve o anki (x,y) değerleri çoğullama çözücünün (demultiplexer) çıkışından işleme sokulmadan doğrudan yol seçici (multiplexer) girişine transfer edilir. Hedef açıya ulaşıldığı anda yani z_i residue açısı, $Rref$ referans değerinin altına düştüğünde, rotasyona uğramış yeni vektörün x ve y koordinat bileşenleri “CORDIC İşlemci” birimi çıkışında doğrudan elde edilir.

Koushik’in yeni CORDIC yönteminde [10] girişi açısının değerine bağlı olarak 1 ve $1/\sqrt{2}$ ölçek faktörü kullanılmaktadır. Önerilen CORDIC modelinde [12] ise [10]’un aksine K ölçek sabiti algoritmada kullanılmamaktadır. K ölçek faktörünün algoritmada kullanılmayışı tasarımda ilave toplama / öteleme operasyonlarının sayısını azaltmakla birlikte K ölçek faktöründen algoritmanın bağımsız oluşu, iterasyon sayısını ciddi oranda azaltmaktadır.

4.2.4.1.2.1. Yeni Açı Değerlerinin Tespiti İçin Alan/Quadrant Tespit Birimi

Şekil 4-31, önerilen yöntem için kartezyen koordinat uzayında quadrantların alanlara ayrıştırılmasını göstermektedir.



Şekil 4 - 31. Quadrantların alanlara parçalanması

Önerilen algoritmanın ilk quadrantı için rotasyona uğramış yeni açı değerlerinin tespiti sözde kod ile açıklanacak olursa;

```
if (angle < 30^0) then φ = θ {skip 1st iteration; i = 2}
elseif (30^0 ≤ angle < 45^0) then φ = θ {consider 1st iteration; i = 1}
else (45^0 ≤ angle < 90^0) then φ = π/2 − θ {test iteration situation}
        if (φ ≥ 30^0) then φ = θ {consider 1st iteration, i = 1}
        else
        φ = θ {skip 1st iteration, not considered, i = 2}
        endif
endif
endif
```

Şekil 4 - 32. Hedef açı değerinin tespit edilmesini temsil eden sözde kod parçası

Şekil 4-31'de ϕ yakısama açısının değerine bağlı olarak koordinat uzayındaki 16 farklı alan gösterilmiştir. Eğer ϕ açısı Alan B veya Alan C'de tespit edilirse Denklem (4.32) kullanılır. Eğer ϕ açısı Alan A veya Alan D'de tespit edilirse Denklem (4.33) kullanılır. θ açı değeri x ve y koordinat bileşenlerinin yer değiştirip değiştirmeyeceğini (swap edilip edilmeyeceğini) belirler. İlk quadrant için bu durumu açıklayan sözde kod Şekil 4-33'de verilmiştir.

```
if (θ < 45^0) then {φ = θ and (x = cosφ, y = sinφ)}
elseif (45^0 ≤ angle < 90^0) then {φ = π/2 − θ and (x = sinφ, y = cosφ)}
endif
endif
```

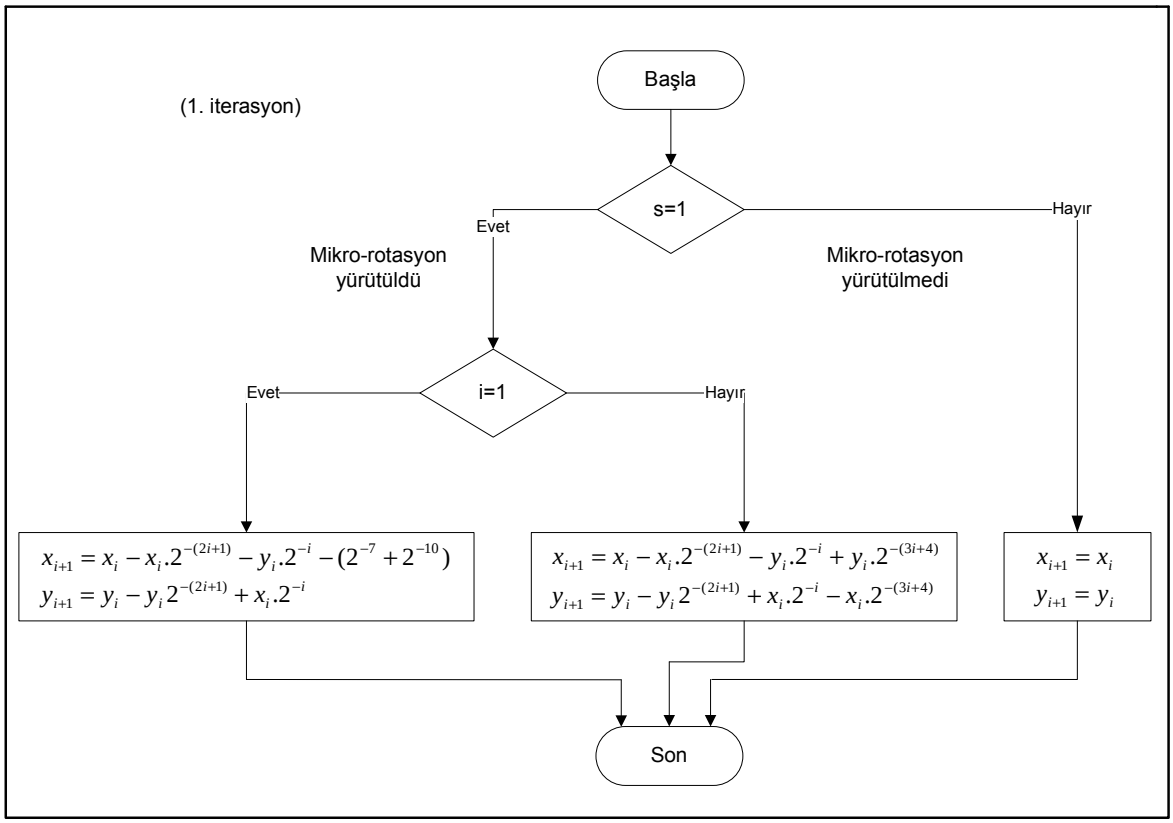
Şekil 4 - 33. x ve y koordinat bileşenlerinin açı değerine göre düzenlenmesini ifade eden sözde kod parçası

Açı ve koordinat bileşenleri diğer quadrantlar için de koordinat sisteminin simetri özelliğinden yararlanılarak hesaplanabilir. Önerilen CORDIC algoritması [12], ilk quadrant için incelenecek olursa, Alan A ve Alan D'deki açı bölgeleri için koordinat değeri hata oranı beklendiği üzere [11] ile aynı olacaktır. Bununla beraber
Alan B ve Alan C'deki açı bölgeleri için hata oranları [11] ile karşılaştırıldığında ciddi bir şekilde düşük olacaktır.

4.2.4.1.2.2. Bir Vektör Rotasyonu İçin x ve y Koordinat Değerlerinin Hesaplanması

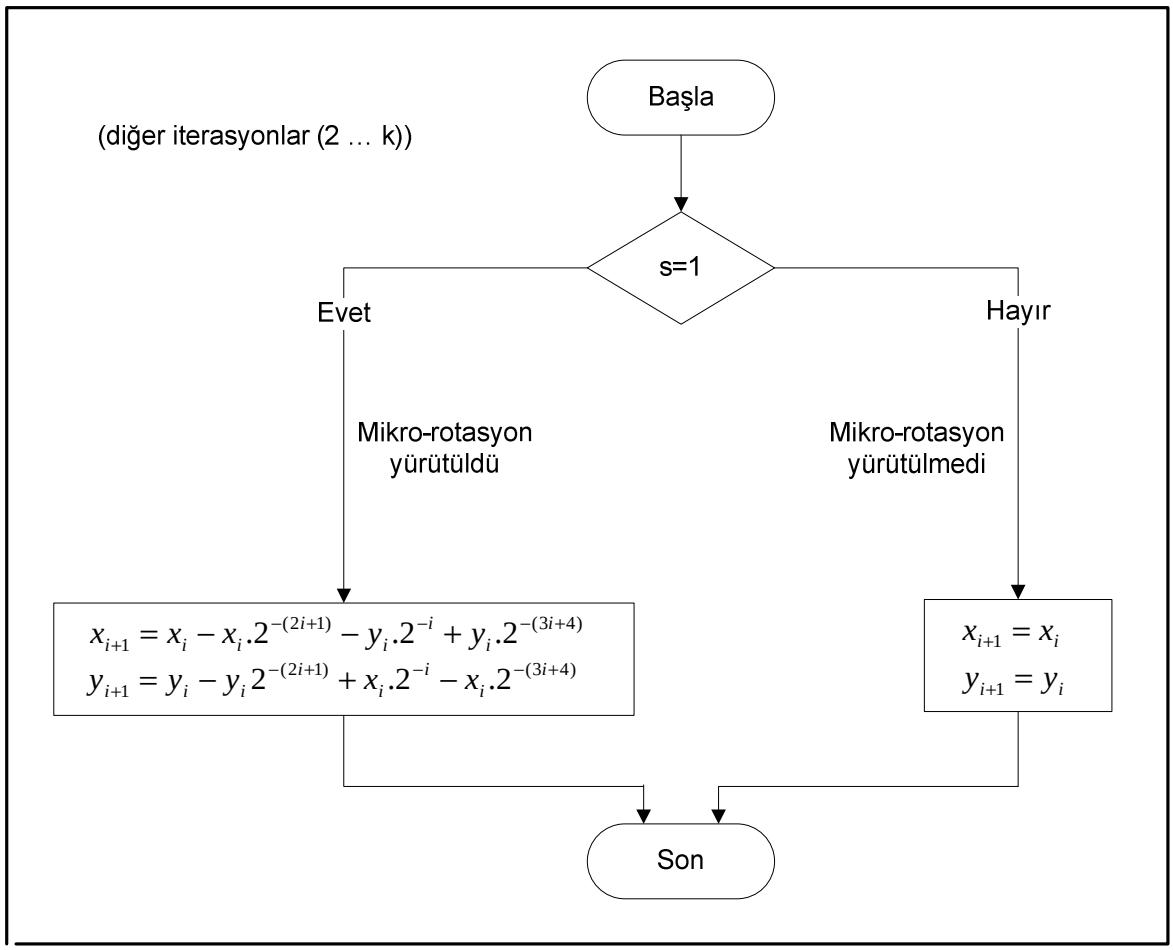
Şekil 4-34'de görüldüğü gibi rotasyon işlemlerinin başlangıcında ilk iterasyon indeksi (i=1) bir defaya mahsus kontrol edilmektedir.

(**s**: rotasyon aktikle/pasifle biti, **i**: iterasyon biti olmak üzere)

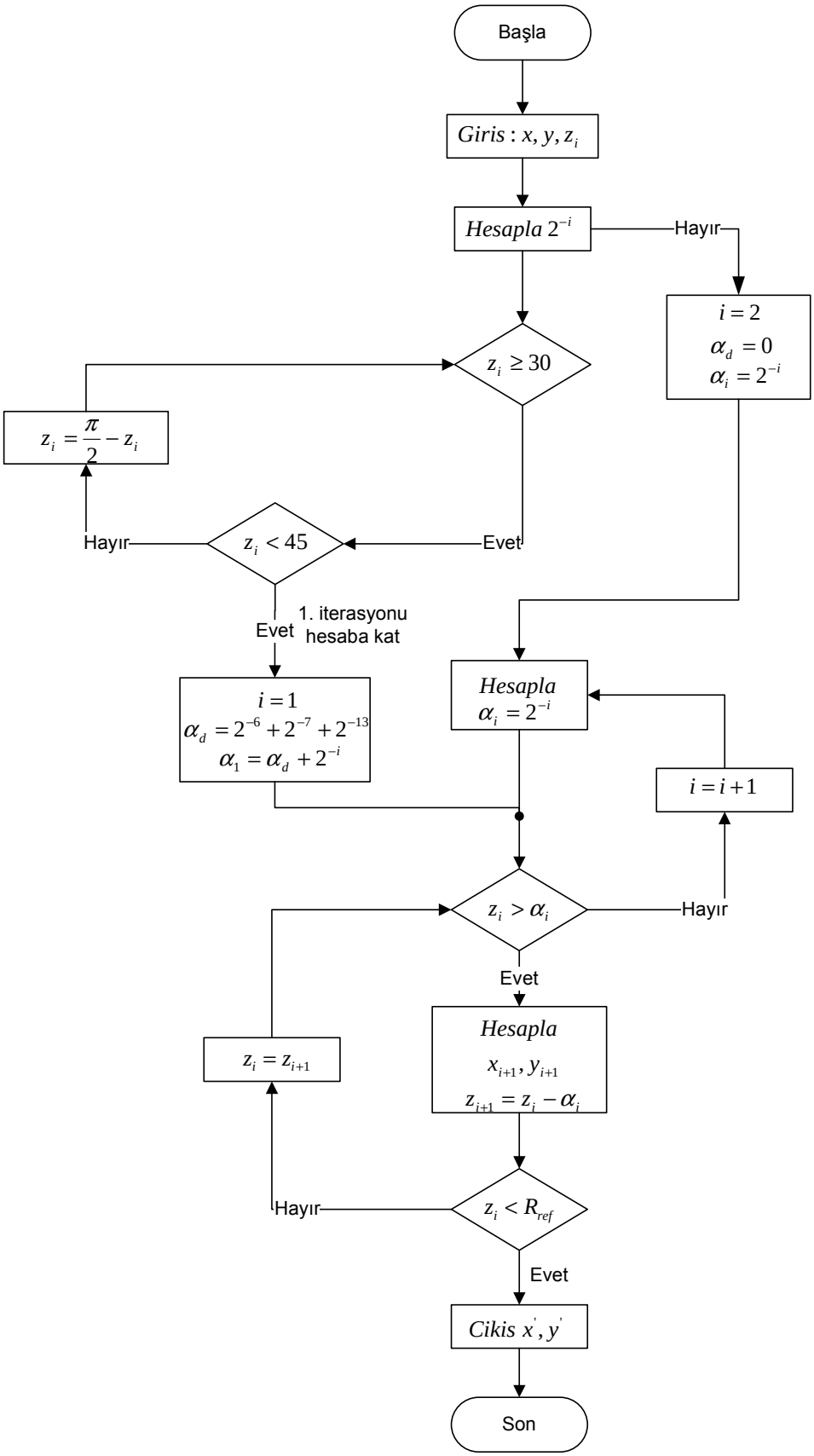


Şekil 4 - 34. Algoritmanın ilk iterasyon için akış diyagramı

İkinci iterasyon adımından (i=2), sonuç koordinat değerleri (x, y) elde edilinceye kadar ki iterasyonlar (i>2) için algoritmanın akış diyagramı Şekil 4-35'de verilmiştir.



Şekil 4 - 35. Algoritmanın ikinci iterasyon adımından itibaren çalışmasını ifade eden akış diyagramı



Şekil 4 - 36. [12] algoritması için akış diyagramı

Şekil 4-36'da [12]'de açıklanan algoritmanın çalışmasını ifade eden akış diyagramı gösterilmiştir.

Her bir iterasyonda α_i mikro rotasyon açısı ile z_i residue değeri hesaplanır ve birbirleriyle karşılaştırılır. Eğer z_i residue değeri, α_i mikro rotasyon açısından küçük ise yeni x ve y değerleri işleme konmaz fakat iterasyon sayısı ($z_i > \alpha_i$) koşulu sağlanıncaya kadar 1 artırılır. Koşul sağlandığı anda yeni x, y ve z_i değerleri hesaplanır. z_i residue değeri ref referans açı değerinden küçük olduğu anda iteratif algoritma sonlanır. En son durumdaki iterasyon sayısı algoritmanın istenen θ hedef açısı kadarlık rotasyonu yapabilmesi için ihtiyaç duyulan iterasyon sayısını vermektedir.

Algoritma düşük sayıda iterasyona sahip ise algoritma sonuca hızlı bir şekilde yakınsıyor anlamı çıkarılabilir, aksi durumda ise algoritmanın yavaş çalıştığı sonucu çıkarılabilir. Önerilen CORDIC algoritması da tıpkı [9] ve [11] gibi seçim algoritmasını kullanmaktadır. Bu sebeple, ref referans açı değeri çok dikkatli bir şekilde seçilmelidir. Referans açı değerinin seçimine göre donanım ve hız açısından iki önemli durum ortaya çıkmaktadır.

1. **Maksimum doğruluk, maksimum donanım, düşük hız, yüksek güç tüketimi**

   ref referans açı değeri çok küçük seçilirse, algoritma yüksek doğrulukta sonuçlara yüksek iterasyon sayısı ile ulaşacaktır.

2. **Minimum doğruluk (yüksek hata), minimum donanım, yüksek hız, düşük güç tüketimi**

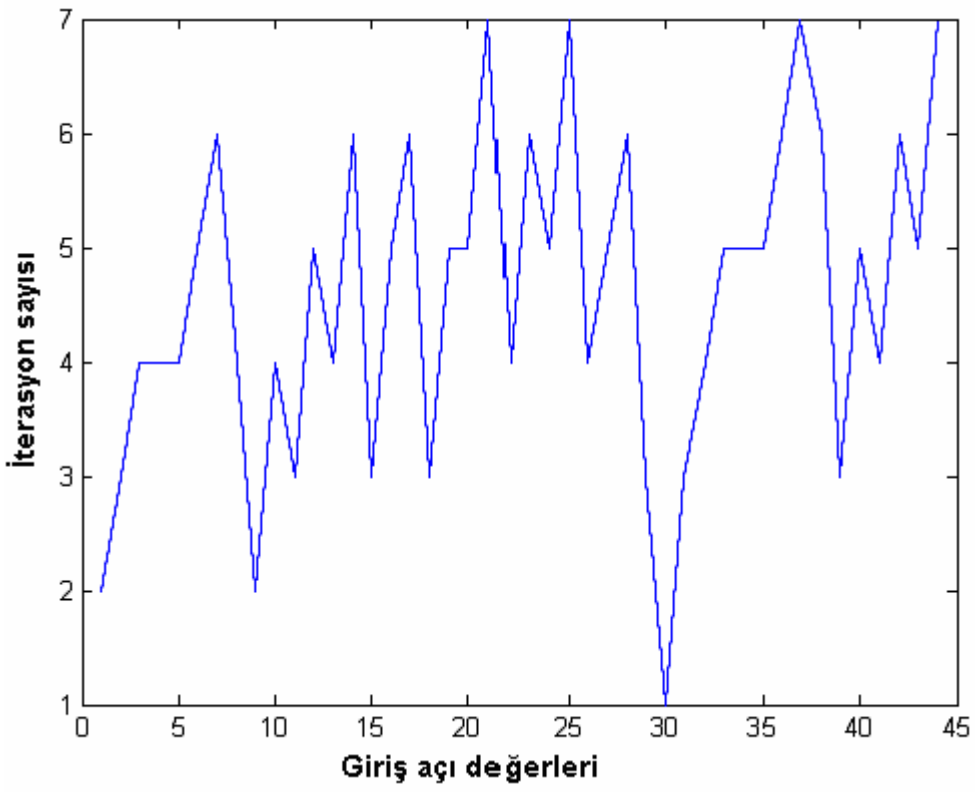
   ref referans açı değeri çok büyük seçilecek olursa, algoritma büyük hata değerlerine sahip sonuçlara düşük iterasyon sayısı ile ulaşacaktır.

Donanım ve hız, algoritmanın performans karakteristiğini gösteren en önemli parametrelerin başında gelmekte olup, önerilen CORDIC modeli [12] adından da anlaşılacağı üzere, yüksek doğrulukta sonuçlara minimum iterasyonda ulaşmayı

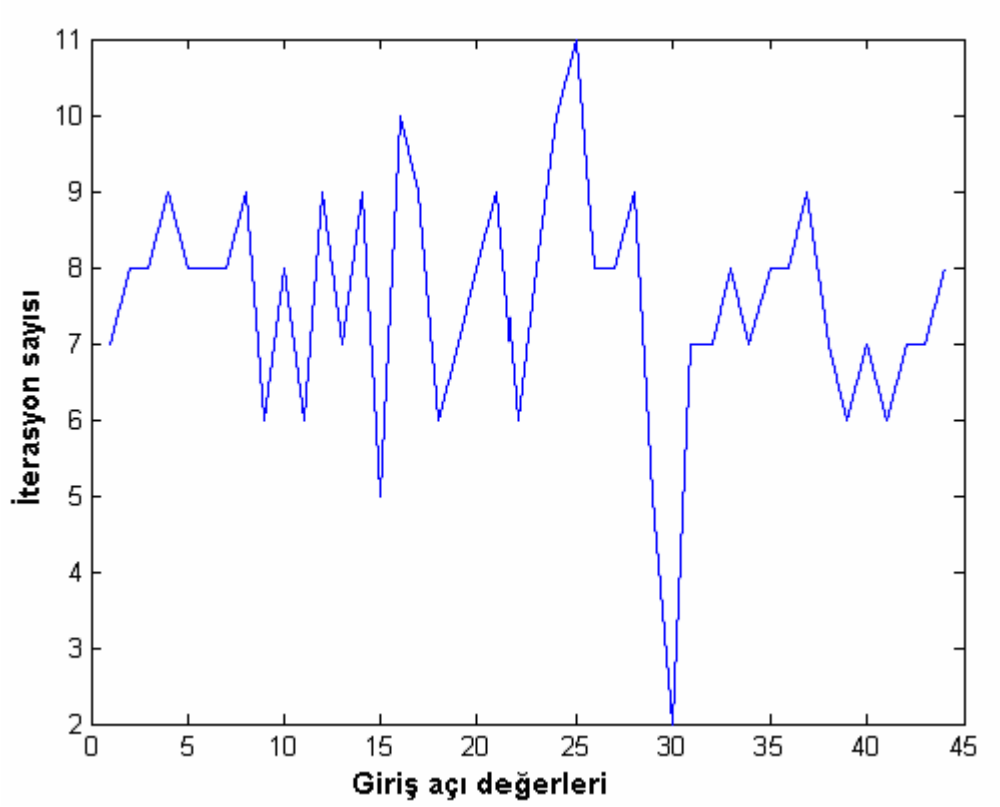
hedeflediğinden donanım ve hız açısından çok sayıda CORDIC algoritmasına göre başarılı bir algoritmadır diyebiliriz.

Bu bölümde verilen tüm sonuçlar “GUICORDIC” MATLAB programı kullanılarak elde edilmiştir.

Maksimum n=10 iterasyon adımı için önerilen algoritma ref referans açı değeri 2^{-10} olarak seçildiği taktirde, “GUICORDIC” MATLAB programı simüle edildiğinde (0-45) derece aralığında algoritmanın yürütülebilmesi için ihtiyaç duyulan iterasyon sayılarının dağılımı Şekil 4-37’de verilmiştir. Maksimum n=16 iterasyon adımı için önerilen algoritma ref referans açı değeri 2^{-16} olarak seçildiği taktirde ihtiyaç duyulan iterasyon sayılarının dağılımı Şekil 4-38’de verilmiştir.



Şekil 4 - 37. n=10 için (0-45) derece aralığındaki açı girişleri için ihtiyaç duyulan iterasyon sayısı dağılımı



Şekil 4 - 38. n=16 için (0-45) derece aralığındaki açı girişleri için ihtiyaç duyulan iterasyon sayısı dağılımı

Tablo 4-17'de bu bölümde incelenen CORDIC algoritmalarının mimari yapılarının karşılaştırması yapılmıştır.

Algoritma Adı	Rotasyonel seçim	Temel rotasyonların yürütülmesi	Arama algoritması (search algorithm)	Gerekli iterasyon sayısı	Ölçek faktörü kullanımı	Desteklenen Maksimum Kelime Uzunluğu
Klasik CORDIC	{-1, 1}	Tamamı	Kullanılmıyor	b (kelime uzunluğu) Sabit	Sabit	32-bit
Düşük Güç Tüketen CORDIC Çekirdeği[11]	{0, 1}	Seçmeli	Kullanılıyor	N, değişken N < b	Yok	32-bit
Gerçekte Ölçekleme Bağımsız Rotasyonel CORDIC [10]	{0, 1}	Seçmeli	Kullanılmıyor	N, değişken N < b	1 veya $1/\sqrt{2}$ (sabit)	20-bit
Yüksek Doğruluklu Düşük İterasyonlu CORDIC İşlemci[12]	{0, 1}	Seçmeli	Kullanılıyor	N, değişken N < b	Yok	32-bit

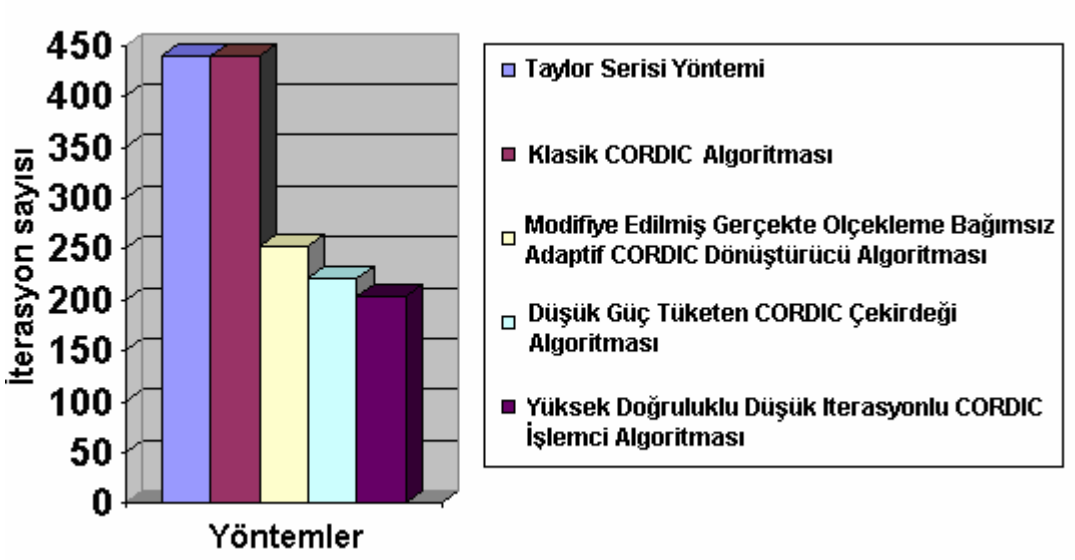
Tablo 4 - 17. Algoritmaların Mimari Karşılaştırması

5. CORDIC ALGORİTMALARI UYGULAMA SONUÇLARI

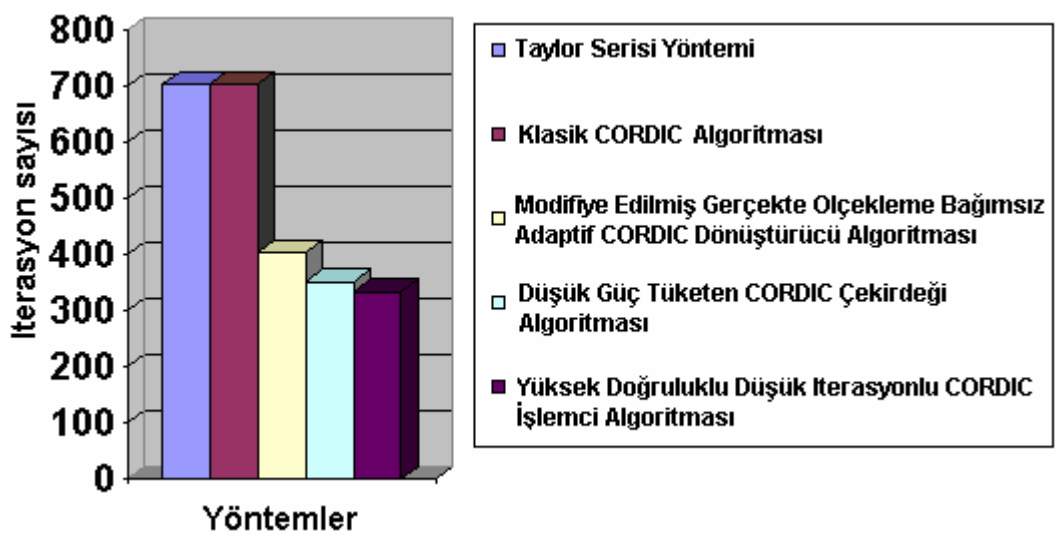
5.1. MATLAB Yazılımı Test Sonuçları ve Karşılaştırılması

5.1.1. İterasyon Sayılarına Göre

Bu tez çalışmasının bir ürünü olan CORDIC algoritması [12], maksimum 10 iterasyon adımı için Klasik CORDIC algoritmasına [1] göre %54.09, Zhang'in CORDIC modeline [11] göre %8.59 ve Koushik'in yeni yaklaşımına [10] göre %19.8 daha düşük iterasyonda sonuca ulaşabilmektedir. Tüm algoritmalar [10], [11], [12] maksimum 16 iterasyon adımı için yürütüldüğünde ise önerilen algoritma [12], Klasik CORDIC algoritmasına [1] göre %52.69, Zhang'in CORDIC modeline [11] göre %5.39 ve Koushik'in yeni yaklaşımına [10] göre %17.57 daha düşük iterasyonda sonuca ulaşabilmektedir. Maksimum n=10 ve n=16 iterasyon adımı için "GUICORDIC" MATLAB programının ürettiği sonuçlar sırasıyla Şekil 5-1 ve Şekil 5-2'de verilmiştir.

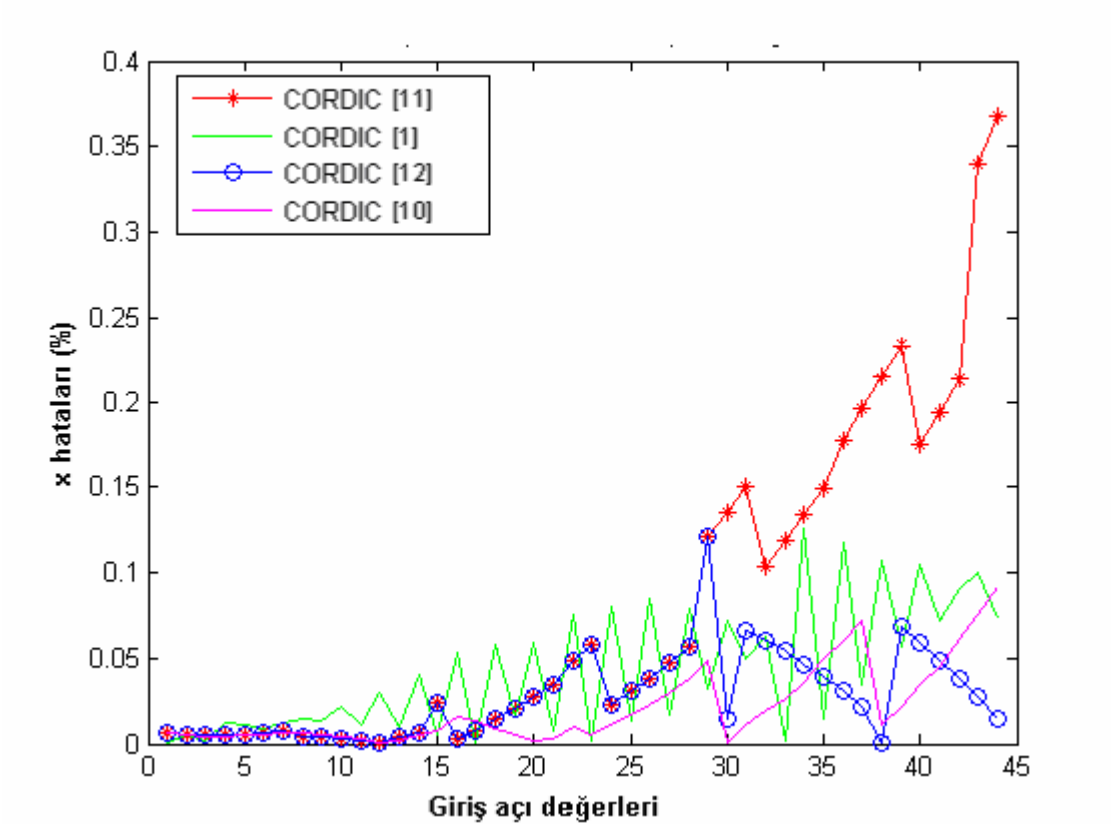


Şekil 5 - 1. n=10 için (0-45) derece açı aralığındaki açı girişleri için farklı algoritmalar yürütüldüğünde ihtiyaç duyulan iterasyon sayısı dağılımının karşılaştırılması

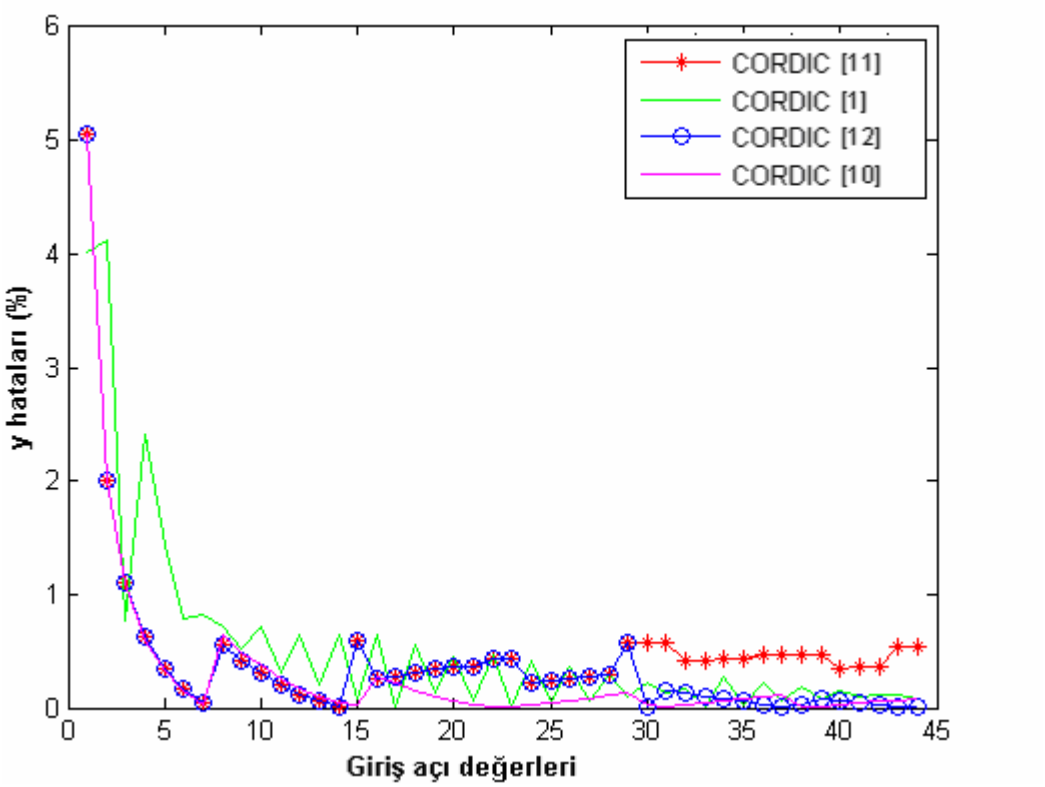


Şekil 5 - 2. n=16 için (0-45) derece açı aralığındaki açı girişleri için farklı algoritmalar yürütüldüğünde ihtiyaç duyulan iterasyon sayısı dağılımının karşılaştırılması

5.1.2. Hata Oranlarına Göre



Şekil 5 - 3. n=10 için (0-45) derece açı aralığındaki açı girişleri için farklı algoritmalar yürütüldüğünde oluşan x koordinat bileşenlerinin hata yüzdelerinin dağılımının karşılaştırılması



Şekil 5 - 4. n=10 için (0-45) derece açı aralığındaki açı girişleri için farklı algoritmalar yürütüldüğünde oluşan y koordinat bileşenlerinin hata yüzdelerinin dağılımının karşılaştırılması

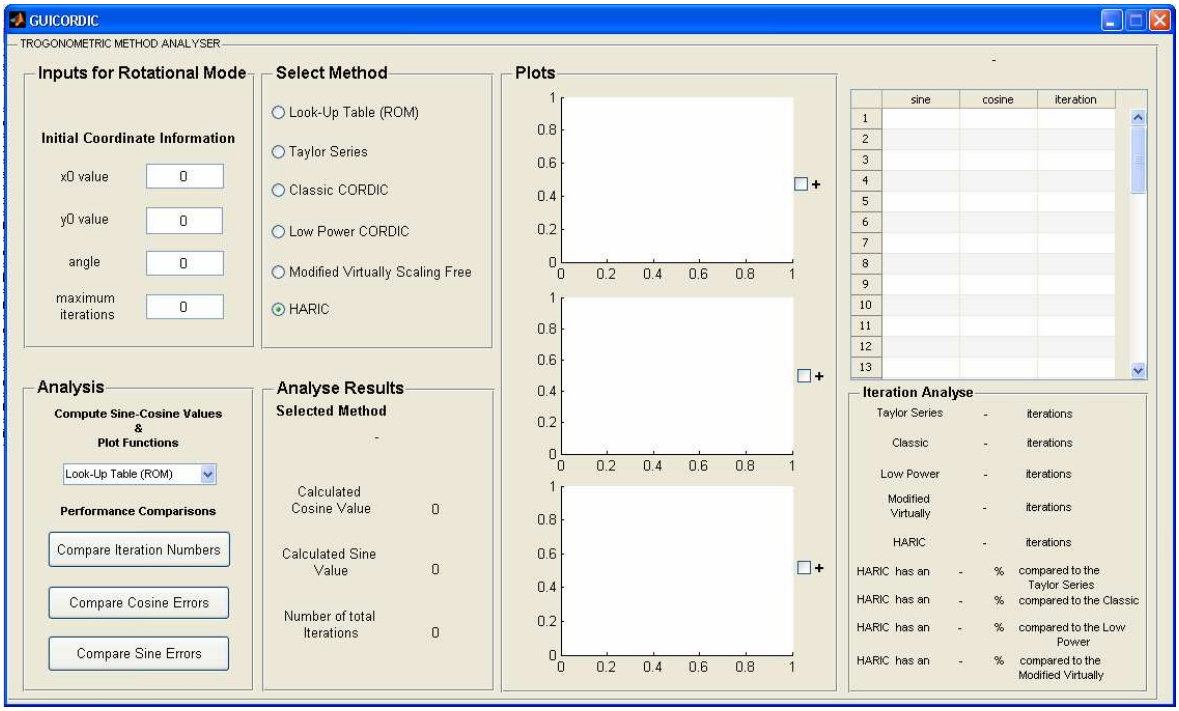
Şekil 5-3 ve Şekil 5-4 'de görüldüğü gibi, önerilen CORDIC algoritması [12], [11]'e göre (0-45) derece aralığı boyunca analiz edildiğinde (30-45) derece açı aralığında daha düşük hata yüzdesinde sonuçlar elde edildiği gözlenmektedir. Bununla beraber (0-29] derece açı aralığında hata yüzdeleri [11] ve [12] için aynıdır. Bununla beraber ilk quadrant boyunca önerilen algoritma [12], [30-60) derece aralığında [11]'e göre iterasyon sayısı ve doğruluk oranları bakımından avantaj sağlamaktadır. Koordinat uzayının simetri özelliğinden yararlanılarak söz konusu avantaj diğer quadrantlarda da aynen sağlanmaktadır.

Önerilen CORDIC algoritması ROM kullanımını temel alan CORDIC yaklaşımları ile vektör koordinat bileşenleri (x,y) açısından karşılaştırıldığında çok yakın doğruluk oranlarına sahip olduğu anlaşılmaktadır.

Sonuçta tez çalışmalarının bir ürünü olan [12]'de mikro rotasyon açısı ve vektör koordinat bileşenleri üzerinde iyileşitirmeler gerçekleştirilerek, algoritma doğruluğu ROM kullanan algoritmalara çok yakın seviyeye getirilmiştir. Bununla beraber algoritmanın yütülebilmesi için ihtiyaç duyulan iterasyon sayısı azaltılarak donanımda kullanılan alanın minimize edilmesi ve algoritma hızının artışı amaçlanmıştır.

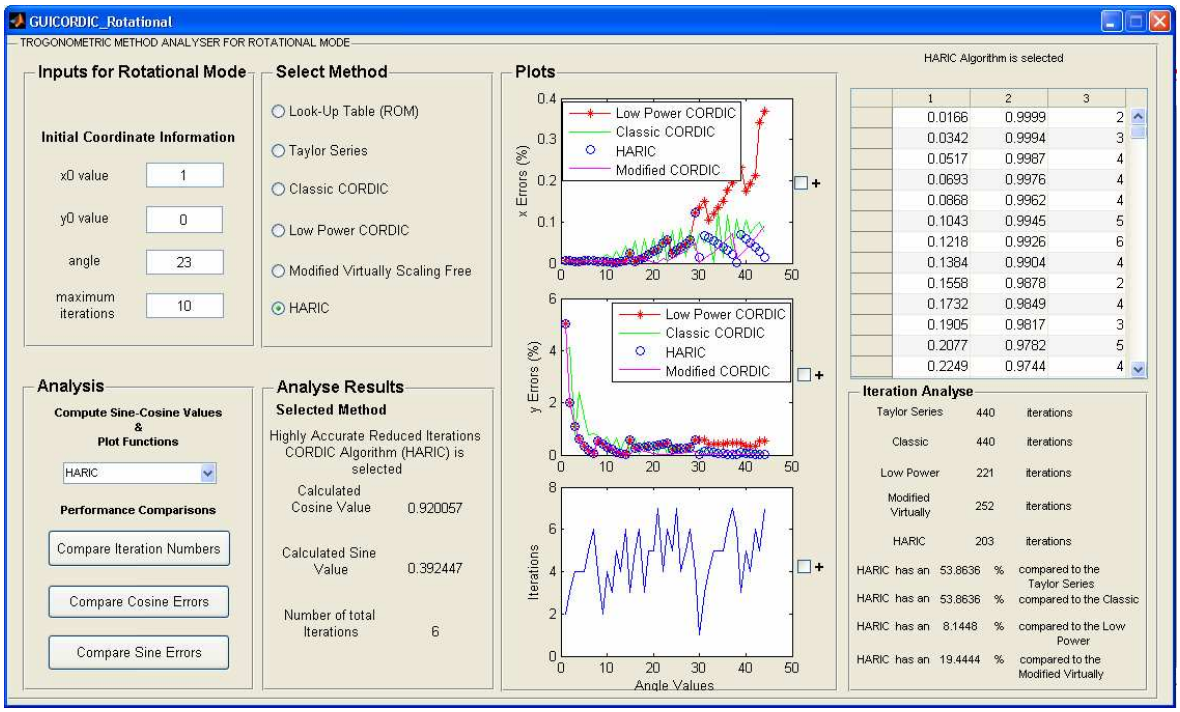
5.2. MATLAB Arayüz Tasarımı ve Uygulaması, GUICORDIC

Şekil 5-5'de gösterilen GUICORDIC grafik kullanıcı arayüzü ekranında, "Inputs for Rotational Mode" sekmesi altında; kullanıcı başlangıç vektör koordinat bileşenlerini, hedef açısını ve maksimum iterasyon sayısını girip, "Select Method" sekmesi altında istenen hesaplama yöntemini seçerek sinüs ve kosinüs değerlerini hesaplayabilir. Hesaplanan sinüs ve kosinüs sonuç değerleri "Analyse Results" sekmesi altında sergilenmektedir.



Şekil 5 - 5. GUICORDIC Grafik Kullanıcı Arayüzü Ekranı

Kullanıcı istediği taktirde analizi koordinat uzayının ilk quadrantı boyunca "Analysis" sekmesi altındaki açılır liste kutucuğundan kullanmak istediği yöntemi seçebilir. Seçim gerçekleştiğinde ekranın sağ üst köşesinde yer alan tabloda ilk quadrant boyunca hedef açılarına karşılık gelen sinüs, kosinüs ve ihtiyaç duyulan iterasyon sayıları Şekil 5-10'da gösterildiği gibi belirecektir. Ayrıca "Plots" sekmesi altında seçilen yönteme ilişkin iterasyon sayılarının ilk quadrant boyunca açı değerlerine göre değişim eğrisi belirecektir.

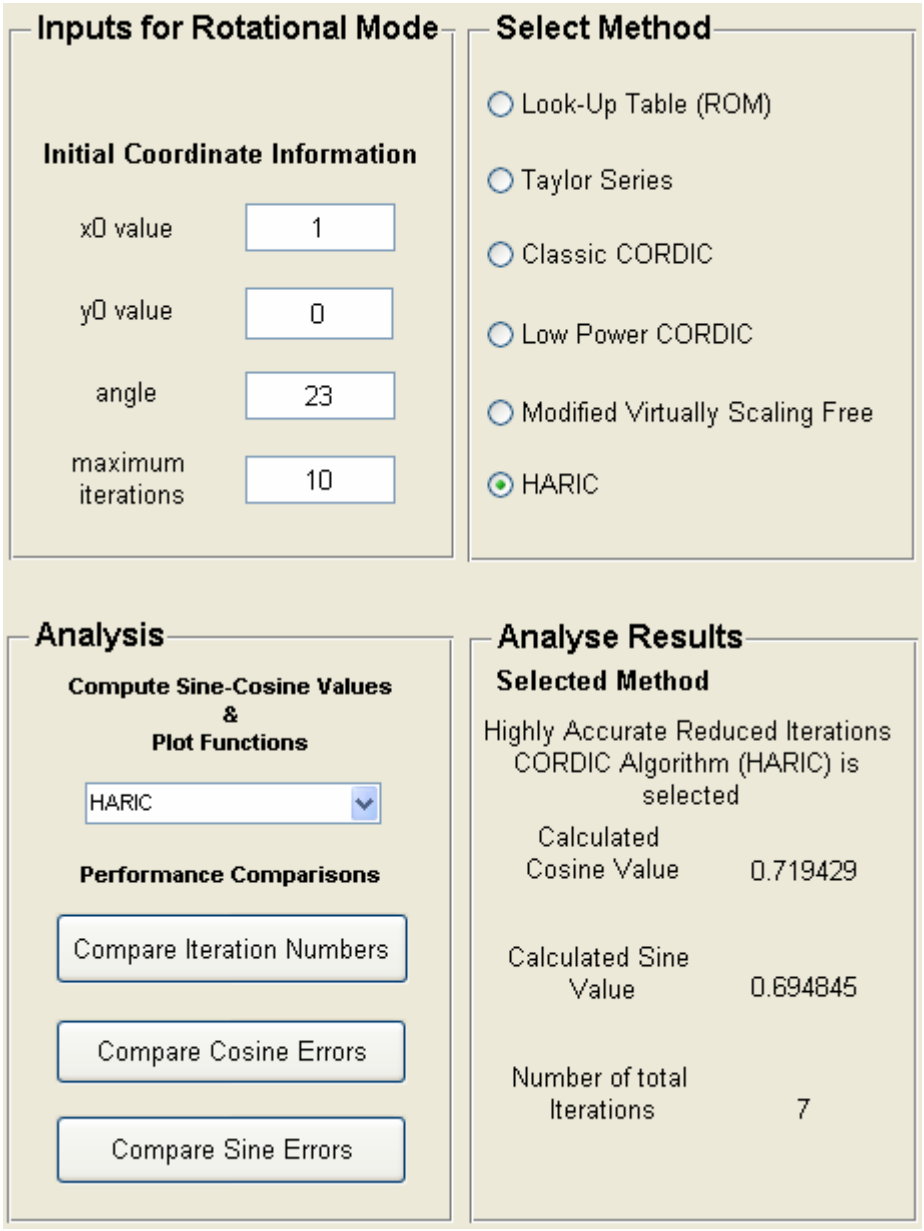


Şekil 5 - 6. GUICORDIC programı ile trigonometrik fonksiyonların performans analizi sonuçlarına ait ekran görüntüsü

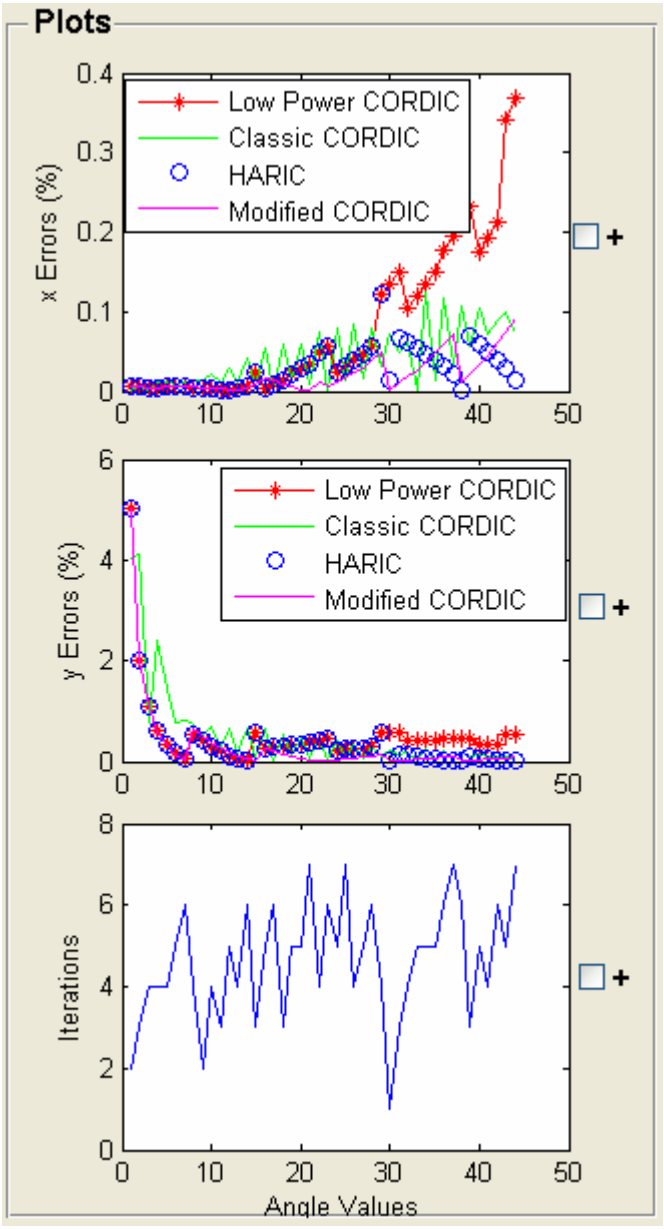
Kullanıcı “Analysis” sekmesi altında “Compare Iteration Numbers” butonuna tıkladığında ilk quadrant boyunca farklı trigonometrik hesaplama yöntemleri için ihtiyaç duyulan iterasyon sayıları analiz sonuçları Şekil 5-6’da görüldüğü gibi ekranın sağ alt kısmındaki “Iteration Analyse” sekmesi altında belirecektir. Şekil 5-9’da ilgili ekran görüntüsü verilmiştir.

Kullanıcı farklı trigonometrik hesaplama yöntemlerini, kosinüs ve sinüs koordinat bileşenlerinin hata oranlarını karşılaştırarak yöntemlerin doğruluk analizini gerçekleştirmek istiyorsa “Analysis” sekmesi altında yer alan “Compute Cosine Errors” ve “Compute Sine Errors” butonlarına tıklamalıdır. Bu durumda “Plots” sekmesi altında yer alan plot ekranlarında, kosinüs ve sinüs bileşenleri için hata oranlarının açı değerlerine göre değişim eğrisi belirecektir. Şekil 5-8’de ilgili gui ekran görüntüsü verilmiştir.

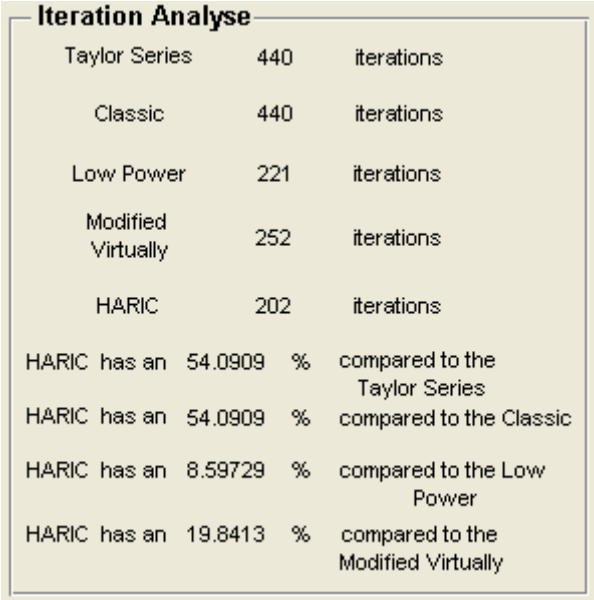
Kullanıcı GUICORDIC grafik arayüzünde ilk quadrant boyunca plot ekranlarında çizdirilen değişim eğrilerini detaylı incelemek istiyorsa ilgili plot ekranının yanında yer alan ‘+’ kutucuğunu seçmelidir, bu şekilde değişim eğrisi daha büyük bir pencerede gösterilecektir.



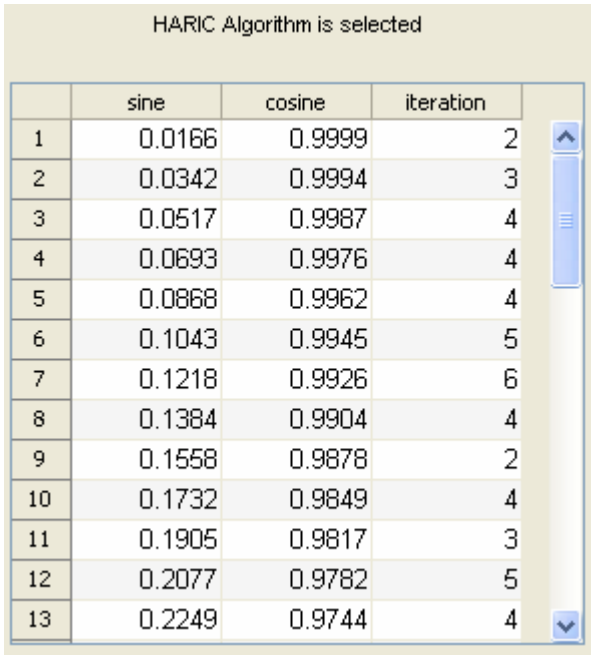
Şekil 5 - 7. GUICORDIC programında rotasyonel mod için trigonometrik fonksiyon yöntemlerinin seçilmesine ve analiz girişlerinin yapılmasına dair ekran görüntüsü



Şekil 5 - 8. GUICORDIC analiz sonuçlarının açı girişlerine gore değişim eğrilerini gösteren “Plots” penceresine ait ekran görüntüsü



Şekil 5 - 9. GUICORDIC programında farklı yöntemler için iterasyon analizi sonuçlarını gösteren “Iteration Analyse” pencere ekran görüntüsü



	sine	cosine	iteration
1	0.0166	0.9999	2
2	0.0342	0.9994	3
3	0.0517	0.9987	4
4	0.0693	0.9976	4
5	0.0868	0.9962	4
6	0.1043	0.9945	5
7	0.1218	0.9926	6
8	0.1384	0.9904	4
9	0.1558	0.9878	2
10	0.1732	0.9849	4
11	0.1905	0.9817	3
12	0.2077	0.9782	5
13	0.2249	0.9744	4

Şekil 5 - 10. GUICORDIC programında ilk quadrant boyunca seçilen yöntem için sinus, kosinüs ve iterasyon sayısını gösteren tablo pencere ekran görüntüsü

5.3. CORDIC Algoritmalarının Donanım Uygulaması

Yapısında donanım çarpıcısı içeren DSP mikroişlemci uygulamalarında, çevrim tablosu yöntemlerinde ve Taylor serisi vb. kuvvet serisi yöntemlerinde CORDIC algoritması kullanımına gerek bulunmamaktadır. Yukarıda bahsedilen teknikler CORDIC algoritmaları ile karşılaştırıldığında daha hızlı çalışmaktadırlar.

Diğer bir yandan donanım çarpıcısı yapısında içermeyen mikrodenetleyici veya FPGA (modern FPGA'ler hariç) vb. donanım sistemlerinde CORDIC algoritması kullanımı önem kazanmaktadır. CORDIC algoritması donanım çarpıcısı bulunmayan programlanabilir işlemci uygulamalarında (mikrodenetleyiciler, FPGA'ler vb..) kaynak kullanımı ve hız bakımından avantaj sağladığı için sıkça kullanılmaktadır.

Bu tez çalışmasında CORDIC algoritmasının yazılımsal uygulaması MATLAB'de donanımsal uygulaması ise FPGA ile gerçeklenmiştir. FPGA uygulaması için Xilinx firmasının Spartan 3-E FPGA geliştirme kiti kullanılmıştır.

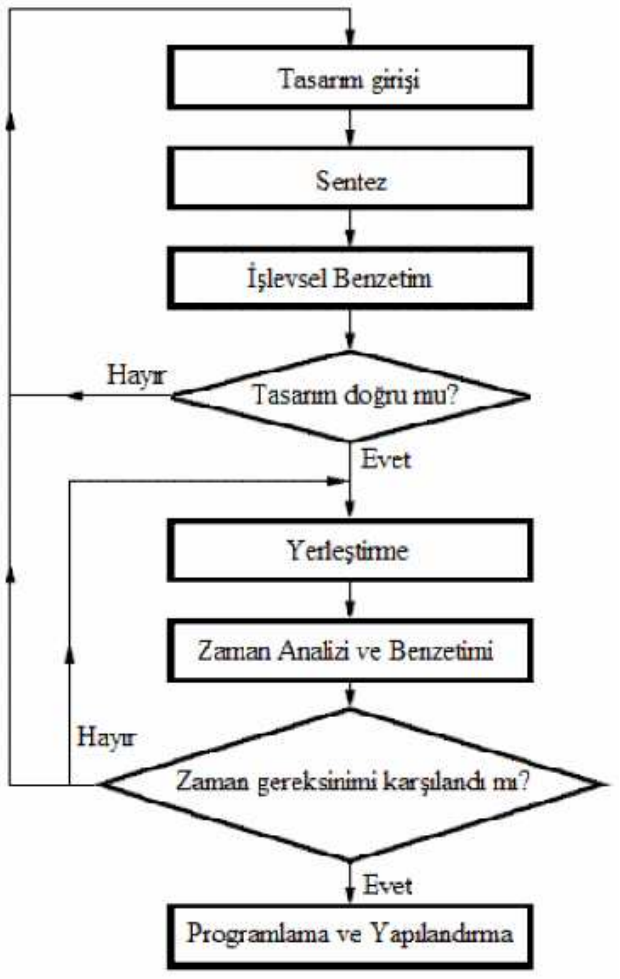
5.3.1. FPGA Tabanlı Tasarımların Avantajları

Günümüz sayısal sistem tasarım teknolojisinde sıklıkla kullanılan yapılar FPGA yongalarıdır. VHDL (Çok Hızlı Tümleşik Devre Donanım Tanımlama Dili) dilinin tasarımcıya sunduğu esneklik ve etkili sistem tasarlama yeteneği sayesinde karmaşık sistemleri çok kısa sürelerde gerçekleştirmek mümkündür. FPGA tabanlı tasarımlar, gerek düşük maliyeti gerekse kısa tasarım süreçleri ile günümüz modern tasarım teknolojisinin vazgeçilmezi olmuşlardır. SoC (System on Chip) çözümler hem PCB (Baskılı Devre Kartı, Printed Circuit Board) alanının büyümesini önlemekte hem de devre güç tüketimlerini azaltmaktadır. Bu avantajların bütününün kullanıldığı platform FPGA yongalarıdır.

5.3.2. FPGA Tasarım Akışı

- Tasarım girişi
- Sentez
- İşlevsel benzetim

- Yerleştirme
- Zaman analizi ve benzetimi
- Programlama ve yapılandırma



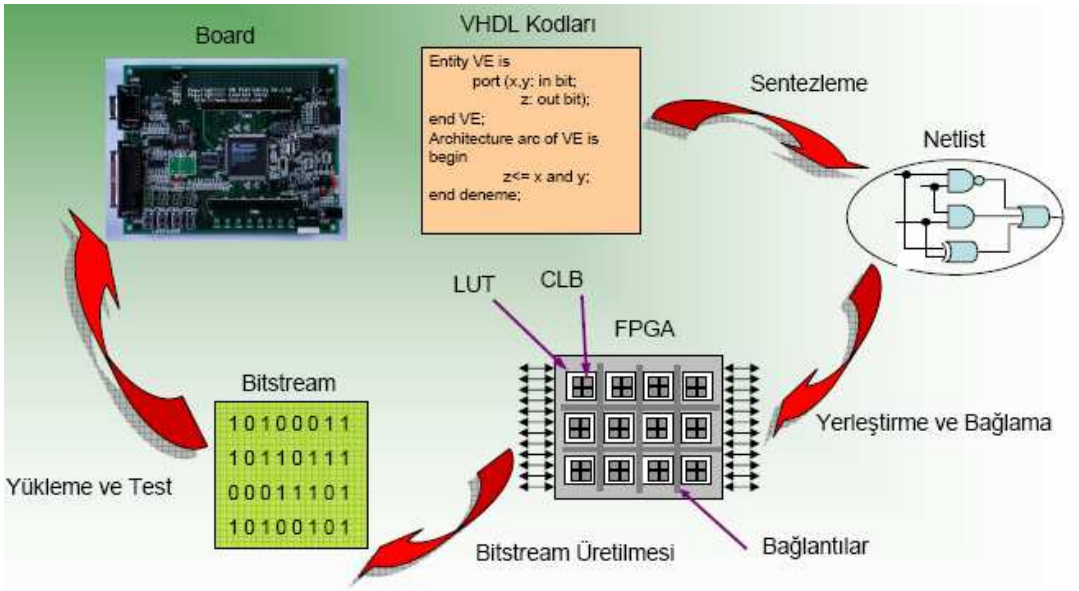
Şekil 5 - 11. FPGA tasarım akışı

5.3.3. FPGA Programlama

Bir FPGA kullanılarak sayısal tasarım yapabilmek için aşağıdaki adımların sırasıyla izlenmesi gerekmekte olup, bu durum Şekil 5-12'de gösterilmiştir.

- Tasarım tanımlanır,
- Netlist oluşturulur,

- Yerleştirme ve bağlantılar oluşturulur,
- Bitstream üretilir,
- Yükleme yapılır.



Şekil 5 - 12. FPGA programlama prosesi

5.3.4. VHDL Tasarım Süreci

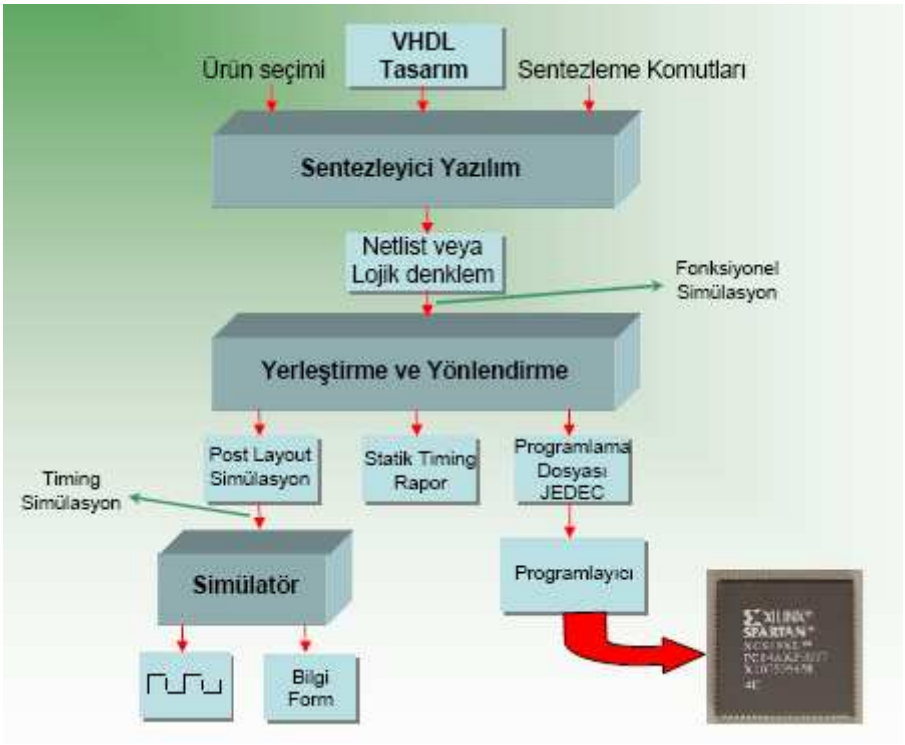
VHDL tasarım süreci şekil 5-13'de ana hatlarıyla gösterilmiştir. VHDL tasarım sürecini çok detaylı bir şekilde incelemek yerine bu bölümde tasarım sürecinin önemli bir adımı olan sentezleme prosesi hakkında bilgi verilecektir.

5.3.4.1. Sentezleme

Sentezleme, tasarım tanımlamasının bir seviyeden daha düşük seviyedeki başka bir seviyeye çevrilmesi olarak tanımlanabilir. Bu çevirme işlemi, C programla dilinde, yüksek seviyeli dilin makina diline çevrilmesine benzetilebilir. Sentezleme programına HDL tanımlama verilir, sentezlenmiş devrede bulunması gereken gecikme ve alan özellikleri programa girilir. Verilen özelliklere sahip devre, sentezleme programı ile, tasarımcı tarafından belirtilen teknoloji kütüphanesindeki elemanlar kullanılarak elde edilir. Sentezleme sonuçları, optimal devre şeması, sentezleme sonucunda elde edilen devreden beklenen performans ve

sentezlenmiş devrenin kapladığı alandır. Aşağıda davranışsal sentezleme, RTL sentezleme ve lojik sentezleme anlatılmıştır.

Davranışsal sentezleme, C programına benzer, algoritmik olarak yazılmış tanımlamanın RTL tanımlamaya çevrilmesidir. RTL'deki tanımlama; veri yolları, bellek elemanları ve kontrol birimlerini içerir. RTL sentezleme, saklayıcı iletişim fonksiyonlarından ardışıl bir devre elde etmek üzere devre şemasının oluşturulmasıdır. Lojik sentezleme, Boole fonksiyonlarının kombinezonsal devre elemanlarıyla gerçeklenmesidir.



Şekil 5 - 13. VHDL akış diyagramı

5.3.5. Farklı CORDIC Algoritmalarının FPGA Uygulamaları

5.3.5.1. Klasik CORDIC Algoritması

5.3.5.1.1 Sistemin Çalışması

Asenkron reset sinyali sistemi başlangıç konumuna almak için kullanılır.

Reset sinyali uygulandığı zaman:

x ve y sinyalleri aşağıdaki başlangıç değerlerine atanır.

x <= 1000000000000000 (işaretsiz, 16-bit)

y <= 0000000000000000 (işaretsiz, 16-bit)

z sinyali ise giriş açısı olarak atanır. (işaretli, 17-bit)

Her bir saat geçişini sayan sayaç indeksi 0'a ayarlanır.

Saat çevriminin her yükselen kenarında ilk 16 saat çevrimi için CORDIC algoritması yürütülür.

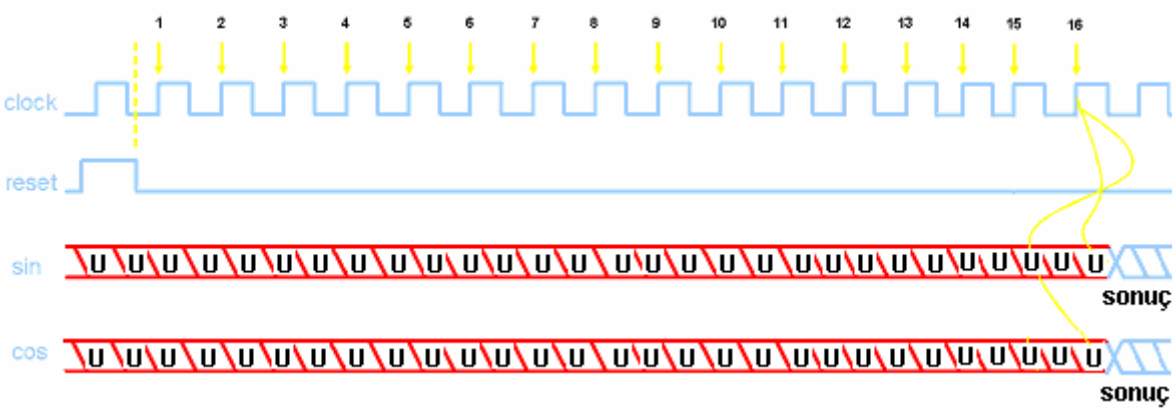
16 saat çevrimi sonrasında sinüs ve cosinüs çıkış değerleri güncellenir.

5.3.5.1.2 Durum makinesı Kod Bloğu (State Machine Code Block)

```
PROCESS( clock, reset )
BEGIN
        IF reset = '1' THEN
                x       <=      "1000000000000000";
                y       <=      "0000000000000000";
                z       <=       angle;
                sayac   <=      "0000";

        ELSIF clock'EVENT AND clock = '1' THEN

              IF sayac = "10001"  THEN
                cosinus <= x;
                sinus   <= y;

              ELSE
                x       <=      x_temp;
                y       <=      y_temp;
                z       <=      z_temp;
                sayac   <=      sayac + 1;
              END IF;
        END IF;
END PROCESS;
```

5.3.5.1.3 Sistem Zamanlaması

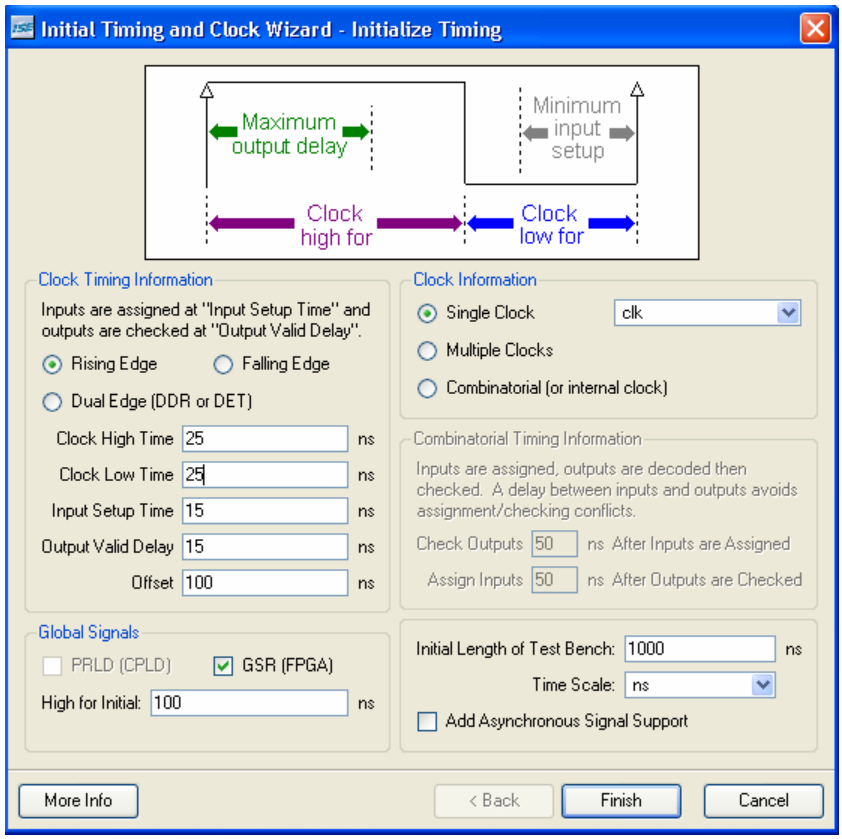
CORDIC, sinüs ve cosinüs sonuçlarını üretmeden önce öteleme birimi (shifter) sonuca ulaşabilmek için 16 saat çevrimine ihtiyaç duymaktadır. Şekil 5-14'de ifade edilen 'U', sonuç elde edilmeden önceki iterasyonlarda "sin" ve "cos" değişkenlerinin tanımsız (undefined) olduğunu belirtmek için kullanılmıştır



Şekil 5 - 14. Klasik CORDIC için sistem zamanlaması

5.3.5.1.4 Frekans

Simülasyon [10], [11], [12] algoritmalarının fonksiyonel doğruluğunu kontrol etmek için 20 MHz'de gerçeklenmiştir. Bu amaçla Xilinx programında VHDL simülasyonu yapabilmek için oluşturulan "Testbench Waveform" kaynağı seçili iken açılan "Initial Timing and Clock Wizard" penceresinde "Clock High Time" ve "Clock Low Time" parametre girdileri 25 ns'ye ayarlanmıştır. (1/50 ns = 20 MHz) Ayrıca "Single Clock" modu seçilmiştir. Bu durum Şekil 5-15'de gösterilmiştir.



Şekil 5 - 15. Xilinx simülasyon zamanlama ve saat çevrimi ayar penceresi

5.3.5.1.5 Simülasyon

5.3.5.1.5.1Sinüs ve Kosinüs Hesaplamaları

Port	Genişlik	Yön	Tanımlama
CLK	1	Giriş	Sistem saati
RESET	1	Giriş	Asenkron reset
Xin	16	Giriş	X girişi
Yin	16	Giriş	Y girişi
Zin	17	Giriş	Açı girişi
Sine	16	Çıkış	Sinüs çıkışı
Cosine	16	Çıkış	Kosinüs çıkışı

Tablo 5 - 1. Sinüs ve kosinüs giriş-çıkış port tanımları

5.3.5.1.6 Hata Analizi

[10], [11], [12] algoritmalarının hata oranları (5.1) ve (5.2) nolu denklemler göz önüne alınarak hesaplanmaktadır.

$$Hata(\text{sinüs(açi)}) = \frac{Simülasyon - Gerçek}{Gerçek} * \ \%100 \tag{5.1}$$

$$Hata(\text{cosinüs(açi)}) = \frac{Simülasyon - Gerçek}{Gerçek} * \ \%100 \tag{5.2}$$

5.3.5.1.6.1Uygulama Sonuçları

Derece		CORDIC [1]	
(onluk)	(16-bit)	Sinüs	Kosinüs
1	572	563	30268
2	1144	1091	30255
3	1716	1617	30234
4	2288	2144	30199
5	2860	2854	32638
6	3432	3421	32586
7	4004	3990	32520
8	4576	4556	32444
9	5148	5121	32362
10	5720	5686	32264
11	6292	6248	32160
12	6862	6808	32046
13	7434	7365	31924
14	8006	7921	31790
15	8578	8474	31646
16	9150	9028	31491
17	9722	9573	31334
18	10294	10123	31160
19	10866	10667	30976
20	11438	11208	30787
21	12010	11741	30586
22	12582	12272	30374

Tablo 5 - 2. Klasik CORDIC sonuçları (1-22) dereceleri için

Derece		CORDIC [1]	
(onluk)	(16-bit)	Sinüs	Kosinüs
23	**13154**	12803	30159
24	**13726**	13324	29933
25	**14298**	13846	29693
26	**14870**	14361	29446
27	**15442**	14871	29191
28	**16014**	15379	28927
29	**16586**	15884	28653
30	**17158**	16381	28371
31	**17730**	16873	28082
32	**18301**	17359	27784
33	**18874**	17845	27476
34	**19444**	18317	27161
35	**20016**	18790	26837
36	**20588**	19253	26507
37	**21160**	19714	26163
38	**21732**	20171	25816
39	**22304**	20619	25461
40	**22876**	21057	25097
41	**23448**	21495	24724
42	**24020**	21924	24346
43	**24592**	22339	23961
44	**25164**	22758	23567

Tablo 5 - 3. Klasik CORDIC sonuçları (23-44) dereceleri için

ÖRNEK-6:

Giriş verileri:

Başlangıç bilgileri: Açı, sinüs, cosinüs olmak üzere;

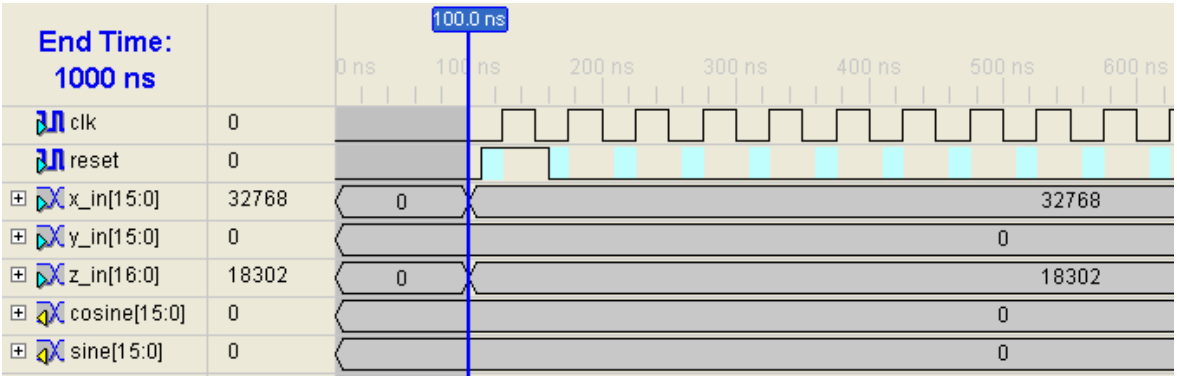
Açı = 00100011101111110 = 18302 (dec) = 32 derece
Sinüs = 0
Cosinüs = 32768 (dec) = 2^15

Çıkış verileri:

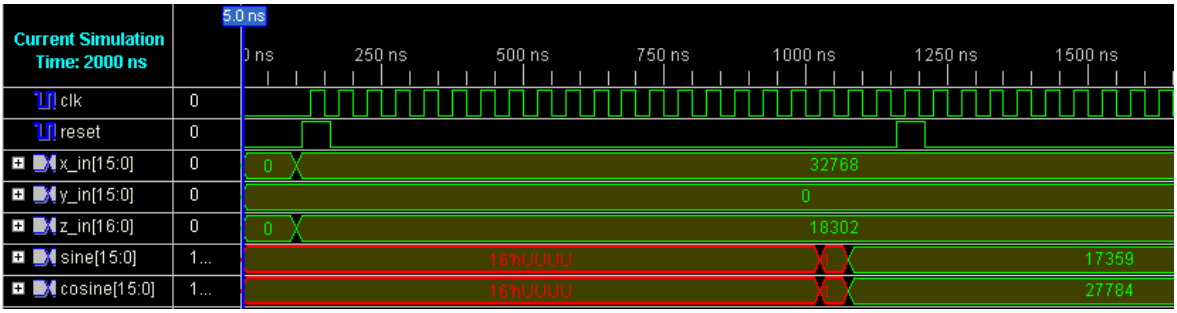
sinüs (Açı) = 0100001111001111 = 17359 (dec)
= 17359 / 2^15
= 0.52975

cosinüs (Açı) = 0110110010001000 = 27784 (dec)
= 27784 / 2^15
= 0.84790

Xilinx ISE Simülatör programında 32 derece giriş açısı için simülasyon Şekil 5-16'ya uygun yürütülecek olursa sonuç Şekil 5-17'deki gibi olacaktır.



Şekil 5 - 16. Klasik CORDIC algoritmasında davranışsal simülasyon modunda 32 derece açısı için simülasyon uyartım penceresi



Şekil 5 - 17. Klasik CORDIC algoritmasında davranışsal simülasyon modunda 32 derece açı girdisi için simülasyon sonuç penceresi

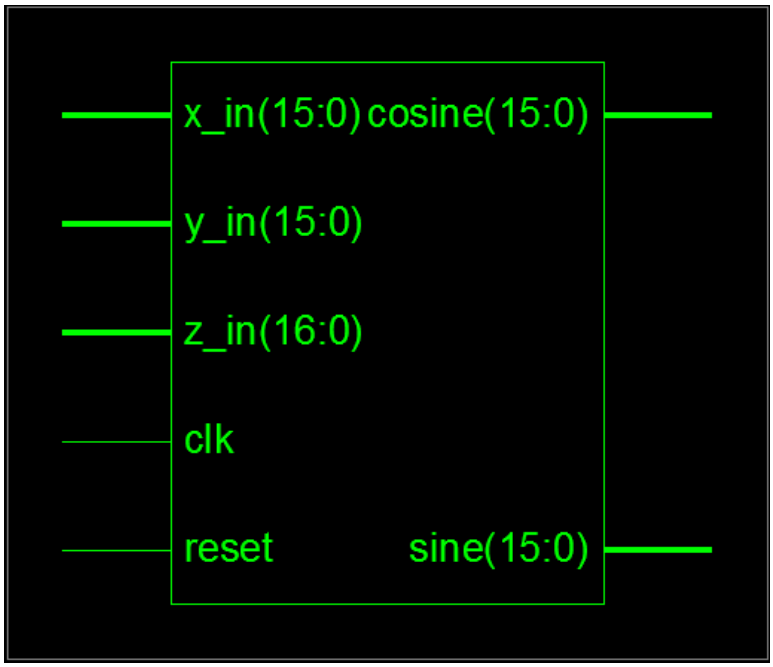
Hata Analizi:

(5.1) ve (5.2) nolu denklemler göz önüne alındığında sünüs ve kosinüs bileşenleri aşağıdaki gibi hesaplanmaktadır.

$$Hata(\text{sinüs}(32)) = \frac{17359 - 17364}{17364} * \ \%100 = \%\ 0.0288$$

$$Hata(\text{cosinüs}(32)) = \frac{27764 - 27789}{27789} * \ \%100 = \%\ 0.0899$$

Xilinx ISE programında sentezleme operasyonu sonrasında Klasik CORDIC algoritması için elde edilmiş olan teknoloji şematik görüntüsü Şekil 5-18'de verilmiş olup, teknoloji şematik görüntülerinin tamamı EK-A'da ayrıntılı olarak verilmiştir.



Şekil 5 - 18. Teknoloji şematik (Klasik CORDIC)

5.3.5.2. Düşük Güç Tüketen CORDIC Çekirdeği Algoritması

Derece		CORDIC [11]	
(onluk)	(16-bit)	Sinüs	Kosinüs
1	572	568	32764
2	1144	1140	32749
3	1716	1711	32727
4	2288	2284	32689
5	2860	2855	32648
6	3432	3424	32593
7	4004	3993	32531
8	4576	4566	32453
9	5148	5132	32367
10	5720	5697	32275
11	6292	6260	32168
12	6862	6818	32057
13	7434	7378	31932
14	8006	7935	31800
15	8578	8532	31649
16	9150	9081	31498
17	9722	9630	31334
18	10294	10178	31158
19	10866	10721	30976
20	11438	11260	30785
21	12010	11794	30585
22	12582	12333	30369
23	13154	12861	30152
24	13726	13386	29924
25	14298	13907	29686
26	14870	14423	29436
27	15442	14938	29178
28	16014	15443	28917

Tablo 5 - 4. Düşük Güç Tüketen CORDIC sonuçları (1-28) dereceleri için

Derece		CORDIC [11]	
(onluk)	(16-bit)	Sinüs	Kosinüs
29	**16586**	15984	28624
30	**17158**	16483	28339
31	**17730**	16975	28047
32	**18301**	17460	27750
33	**18874**	17943	27439
34	**19444**	18419	27125
35	**20016**	18890	26796
36	**20588**	19365	26453
37	**21160**	19818	26117
38	**21732**	20271	25768
39	**22304**	20717	25410
40	**22876**	21160	25044
41	**23448**	21593	24672
42	**24020**	22019	24293
43	**24592**	22446	23896
44	**25164**	22859	23503

Tablo 5 - 5. Düşük Güç Tüketen CORDIC sonuçları (29-44) dereceleri için

ÖRNEK-7:

Giriş verileri:

Başlangıç bilgileri: Açı, sinüs, cosinüs olmak üzere;

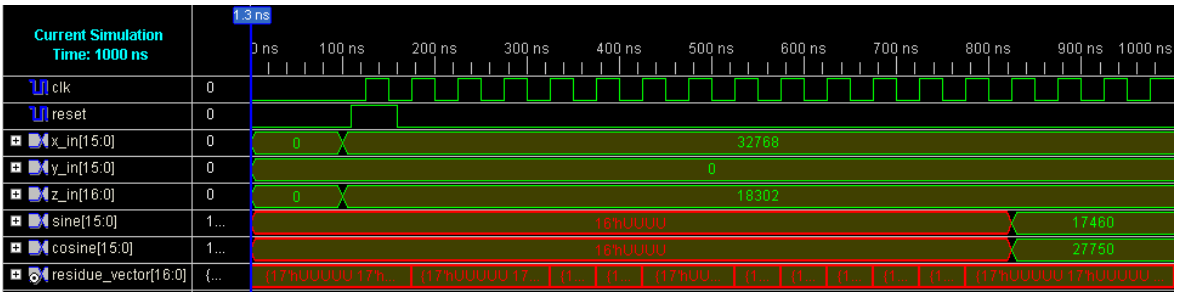
Açı = 00100011101111110 = 18302 (dec) = 32 derece
Sinüs = 0
Cosinüs = 32768 (dec) = 2^15

Çıkış verileri:

sinüs (Açı) = 0100010000110100 = 17460 (dec)
= 17460 / 2^15
= 0.53283

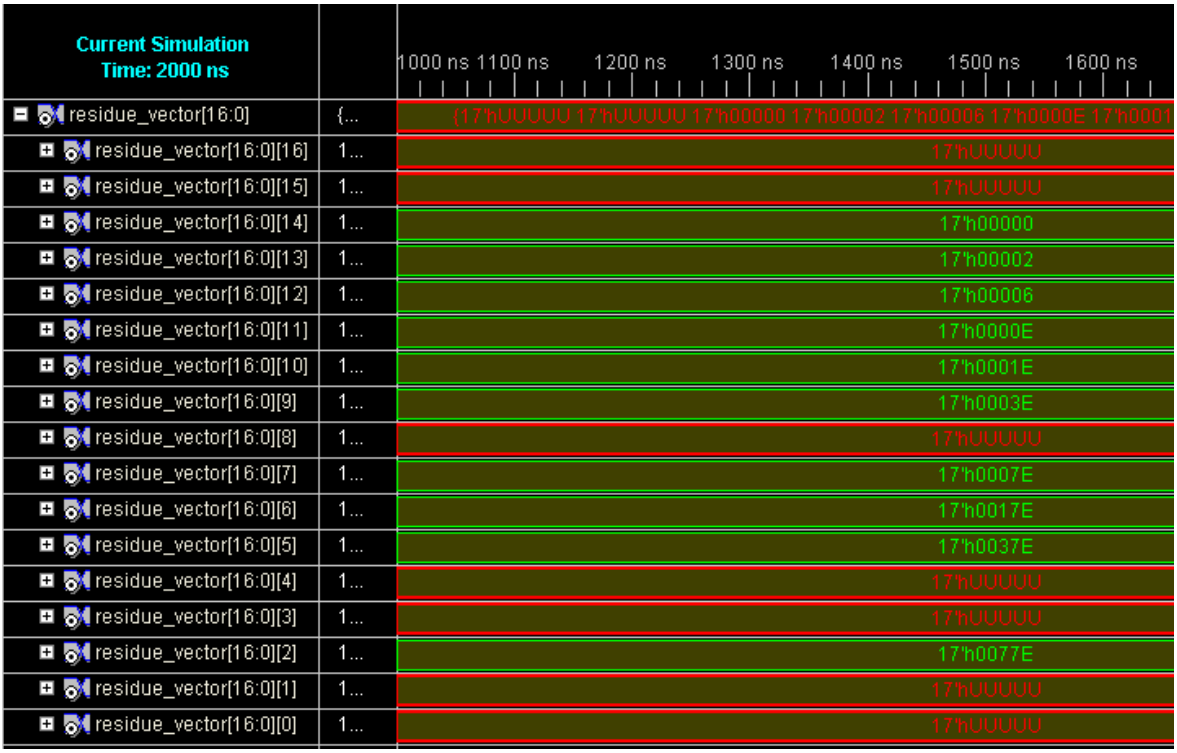
cosinüs (Açı) = 0110110001100110 = 27750 (dec)
= 27750 / 2^15
= 0.84686

Xilinx ISE Simülatör programında 32 derece giriş açısı için simülasyonu yürütülecek olursak sonuç Şekil 5-19'daki gibi olacaktır.



Şekil 5 - 19. Düşük Güç Tüketen CORDIC Çekirdeği algoritmasında davranışsal simülasyon modunda 32 derece açı girdisi için simülasyon sonuç penceresi

32 derece giriş açısı için vektör rotasyonlarının Şekil 5-20'de verildiği gibi yürütülmekte olup, her bir ilgili vektör rotasyonu sonunda güncellenen residue z açı değer bilgisi simülasyon sonuç penceresinde gösterilmiştir.



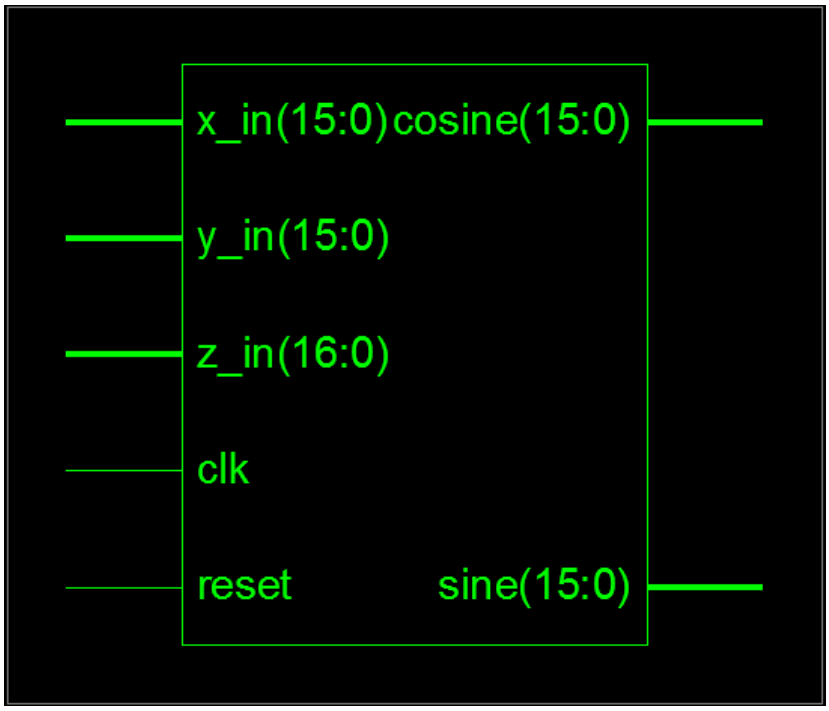
Şekil 5 - 20. Düşük Güç Tüketen CORDIC Çekirdeği algoritmasında davranışsal simülasyon modunda 32 derece açı girdisi için simülasyon sonuç penceresinde residue vektör değerlerinin gösterilmesi

5.3.5.2.1 Hata Analizi

$$Hata(\text{sinüs}(32)) = \frac{17460 - 17364}{17364} * \ \%100 = \%\ 0.5528$$

$$Hata(\text{cosinüs}(32)) = \frac{27750 - 27789}{27789} * \ \%100 = \%\ 0.1403$$

Xilinx ISE programında sentezleme operasyonu sonrasında “Düşük Güç Tüketen CORDIC Çekirdeği” algoritması için elde edilmiş olan teknoloji şematik görüntüsü Şekil 5-21’de verilmiş olup, teknoloji şematik görüntüsü EK-A’da verilmiştir.



Şekil 5 - 21. Teknoloji şematik (Düşük Güç Tüketen CORDIC)

5.3.5.3. HARICP Algoritması

Derece		CORDIC [12]	
(onluk)	(16-bit)	Sinüs	Kosinüs
1	572	568	32764
2	1144	1140	32749
3	1716	1711	32727
4	2288	2284	32689
5	2860	2855	32648
6	3432	3424	32593
7	4004	3993	32531
8	4576	4566	32453
9	5148	5132	32367
10	5720	5697	32275
11	6292	6260	32168
12	6862	6818	32057
13	7434	7378	31932
14	8006	7935	31800
15	8578	8532	31649
16	9150	9081	31498
17	9722	9630	31334
18	10294	10178	31158
19	10866	10721	30976
20	11438	11260	30785
21	12010	11794	30585
22	12582	12333	30369
23	13154	12861	30152
24	13726	13386	29924
25	14298	13907	29686
26	14870	14423	29436
27	15442	14936	29179
28	16014	15443	28917

Tablo 5 - 6. HARICP algoritması sonuçları ve hata oranları (1-28) dereceleri için

Derece		CORDIC [12]	
(onluk)	(16-bit)	Sinüs	Kosinüs
29	16586	16302	28575
30	17158	16385	28383
31	17730	16875	28094
32	18301	17364	27797
33	18874	17846	27490
34	19444	18324	27175
35	20016	18796	26851
36	20588	19261	26519
37	21160	19722	26179
38	21732	20181	25827
39	22304	20629	25470
40	22876	21071	25106
41	23448	21506	24734
42	24020	21933	24355
43	24592	22355	23969
44	25164	22771	23576

Tablo 5 - 7. HARICP algoritması sonuçları ve hata oranları (29-44) dereceleri için

ÖRNEK-8:

Giriş verileri:

Başlangıç bilgileri: Açı, sinüs, cosinüs olmak üzere;

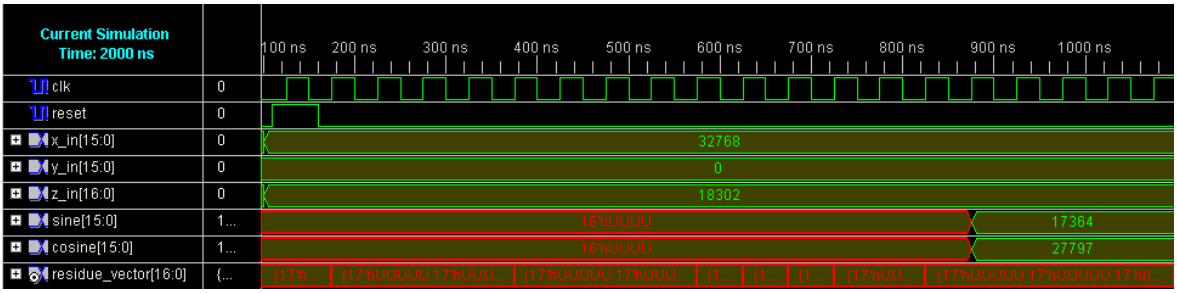
Açı = 00100011101111110 = 18302 (dec) = 32 derece
Sinüs = 0
Cosinüs = 32768 (dec) = 2^15

Çıkış verileri:

sinüs (Açı) = 0100001111010100 = 17364 (dec)
= 17364 / 2^15
= 0.52990

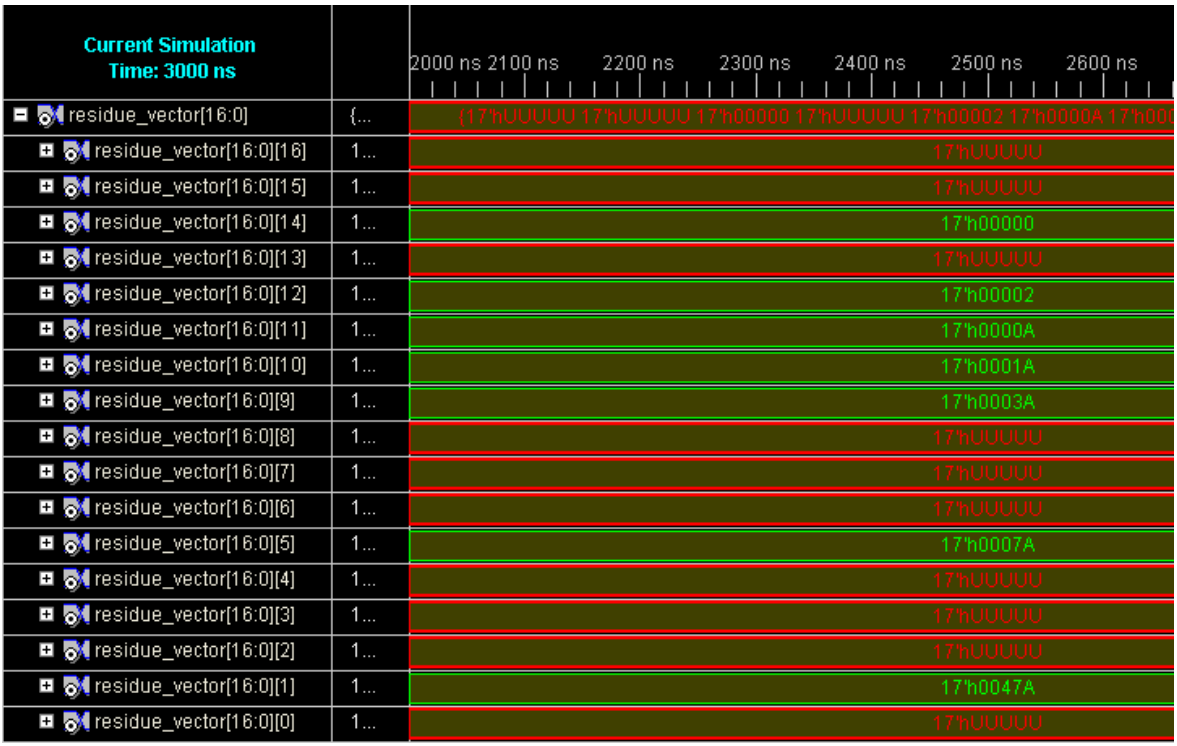
cosinüs (Açı) = 0110110010010101= 27797 (dec)
= 27797 / 2^15
= 0.84829

Xilinx ISE Simülatör programında 32 derece giriş açısı için simülasyon yürütülecek olursa sonuç Şekil 5-22'deki gibi olacaktır.



Şekil 5 - 22. HARICP algoritmasında davranışsal simülasyon modunda 32 derece açı girdisi için simülasyon sonuç penceresi

32 derece giriş açısı için vektör rotasyonlarının Şekil 5-23'de verildiği gibi yürütülmekte olup, her bir ilgili vektör rotasyonu sonunda güncellenen residue z açı değer bilgisi simülasyon sonuç penceresinde gösterilmiştir.



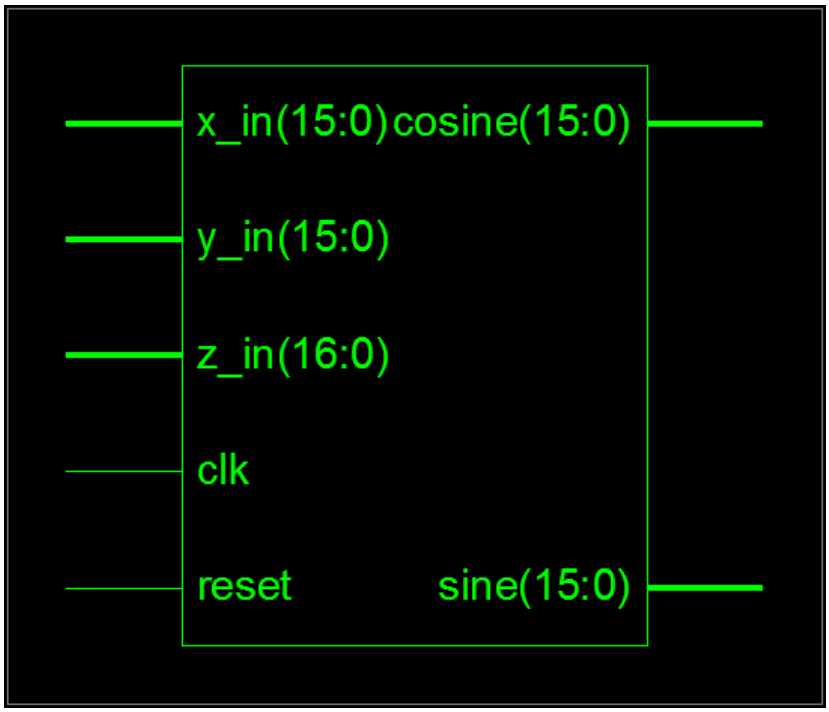
Şekil 5 - 23. HARICP algoritmasında davranışsal simülasyon modunda 32 derece açı girdisi için simülasyon sonuç penceresinde residue vektör değerlerinin gösterilmesi

5.3.5.3.1 Hata Analizi

$$Hata(\text{sinüs}(32)) = \frac{17364 - 17364}{17364} * \ \%100 = \%\ 0.0$$

$$Hata(\text{cosinüs}(32)) = \frac{27797 - 27789}{27789} * \ \%100 = \%\ 0.0287$$

Xilinx ISE programında sentezleme operasyonu sonrasında “Yüksek Doğruluklu Azaltılımış İterasyonlu CORDIC İşlemci” algoritması için elde edilmiş olan teknoloji şematik görüntüsü Şekil 5-24’de verilmiş olup, teknoloji şematik görüntüsü EK-A’da verilmiştir.

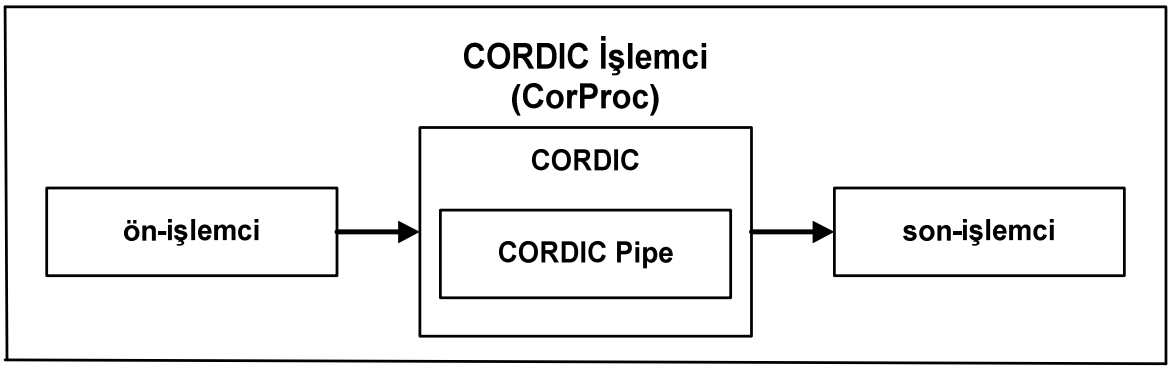


Şekil 5 - 24. Teknoloji şematik (HARICP)

5.3.5.4. Pipeline Çalışan Klasik CORDIC Algoritması

Tüm CORDIC işlemci çekirdekleri üç temel birimden oluşmaktadır. Bunlar; ön-işlemci, son-işlemci ve operasyonların yürütüldüğü CORDIC çekirdeği birimidir.

CORDIC çekirdeği birimi, pipeline olarak çalışan "CordicPipe" blokları kullanılarak oluşturulmuştur. Her bir "CordicPipe" bloğu iterasyon prosesinde bir adımı temsil etmektedir.



Şekil 5 - 25. Pipeline çalışan Klasik CORDIC mimarisi

Pipeline mimarinin dezavantajlı tarafı ise donanımda kullanılan alanın giderek artması, yüksek maliyet ve yüksek güç tüketimidir. Pipeline yapıya sahip bir sistemin yönerge gecikmesi(latency), pipeline olmayan yapıya göre daha fazladır. Bu da sisteme ekstra flip-floplar eklenmesinden kaynaklanır. Pipeline mimarilerde ALU tasarımı daha karmaşıktır.

5.3.5.5. Klasik Ön İşlemci ve Arka İşlemci Birimleri

Klasik CORDIC algoritmasında arktanjant tablosu kullanıldığı için, algoritma sadece [-1, +1] (radyan) aralığında yakınsar. CORDIC algoritması [0, 2π] aralığında yakınsayan giriş açı değerlerini [-1, +1] aralığına manipule etme ihtiyacı duyar. Bu görev "ön-işlemci" tarafından icra edilmektedir.

Son-işlemci CORDIC çekirdeği tarafından üretilen sonuçları uygun quadranta dönüştürür. Son işlemci birimi K ölçekleme faktörü (doğrulama faktörü) operasyonlarından da sorumludur.

5.3.5.6. CORDIC Çekirdeği Birimi

CORDIC çekirdeği birimi CORDIC işlemci çekirdeğinin kalbidir. Bu birim gerçek CORDIC algoritmasının yürütüldüğü kısımdır. Tüm iterasyonlar pipeline yapı kullanılarak paralel yürütülmektedir. Pipeline yapı sayesinde CORDIC çekirdeği her saat çevriminde bir CORDIC operasyonunu yürütmektedir. Bu özellik sayesinde algoritma mümkün olan en yüksek throughput'da sonuçları üretmektedir.

5.3.5.6.1 CORDIC Pipeline Birimi

Her pipe veya iterasyon adımı CORDICPipe çekirdeği tarafından yürütülmektedir. Bu birim her bir iterasyon için arktanjant tablosu ile x, y ve z değerlerinin manipüle eden lojik tasarımı içermektedir.

5.3.5.6.2 Sinüs ve Kosinüs Hesaplamaları

ÖRNEK-9: 30 derecenin sinüs ve kosinüs değerlerinin hesaplanması.

İlk olarak açı hesaplanmak zorundadır. Hesaplamalarda kullanılacak olan veri uzunluğuna karar verilmelidir. Bu örnek için n =16 olduğu varsayılmıştır.

CORDIC algoritması –1 ve +1 radyan değerlerine dönüştürmek için 2π cinsinden bir girişe ihtiyaç duyar.

$$360\ derece = 2^{16}$$

$$1\ derece = \frac{2^{16}}{360}$$

$$30\ derece = \frac{2^{16}}{360}.30 \approx 5461\,(dec) = 1555(hex)$$

CORDIC çekirdeği $z_i = 5461$ için takip eden sinüs ve kosinüs değer hesaplamalarını yapar.

$$Sin: 16380\,(dec) = 3FFC\,(hex)$$
$$Cos: 28381\,(dec) = 6EDD\,(hex)$$

Çıkış [-1, +1] değer aralığında temsil edilir. Bundan dolayı aşağıdaki işlemler yürütülür.

Sinüs hesabı:

$$2^{15} = 1.0$$

$$16380 \equiv \frac{1.0}{2^{15}}.16380 = 0.49987$$

Kosinüs hesabı:

$$2^{15} = 1.0$$

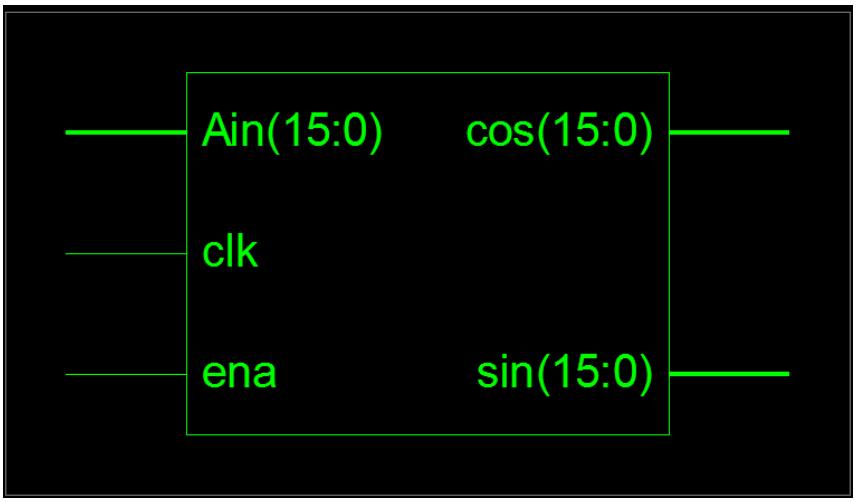
$$28381 \equiv \frac{1.0}{2^{15}}.28381 = 0.8661$$

	0 derece	**30 derece**	**45 derece**	**60 derece**	**90 derece**
Sin (hex)	0x01CC	0x3FFC	0x5A82	0x6EDC	0X8000
Cos (hex)	0x8000	0x6EDD	0x5A83	0x4000	0x01CC
Sin	0.01403	0.49998	0.70709	0.86609	1.00000
Cos	1.00000	0.86612	0.70712	0.50000	0.01403

Tablo 5 - 8 Bazı ortak açı değerleri için sinüs ve kosinüs çıkış değerleri

Yukarıdaki örnekte beklenen sonuçlar sin(30)=0.5 ve cos(30)=0.866 olmalıydı.

16 bit'lik veri girişleri için sinüs ve kosinüs hesaplamasında algoritma çok küçük hata oranlarıyla sonucu bulmuştur. Bu örnek için hata oranı 10^{-4} seviyelerindedir. Xilinx ISE programında sentezleme operasyonu sonrasında "Pipeline Çalışan Klasik CORDIC" algoritması için elde edilmiş olan teknoloji şematik görüntüsü Şekil 5-26'da verilmiş olup, teknoloji şematik görüntüsü EK-A'da verilmiştir.



Şekil 5 - 26. Teknoloji şematik (Pipeline Klasik CORDIC)

5.3.6. FPGA Uygulama Sonuçları ve Karşılaştırılması

5.3.6.1. Sentez Sonuçları

Donanım çarpıcısı bulunmayan programlanabilir mimarilerde; donanımda kullanılan alan, güç tüketimi, yüksek yayılım hızı, düşük maliyet vb. faktörler sebebiyle avantajlı olan CORDIC yöntemi uygulamadaki performans gereksinim önceliğine göre farklı amaçlar için farklı mimarilerde kullanılabilmektedir.

Tablo 5-9'da FPGA donanımında kaynak kullanımı ve oluşan hata oranları farklı CORDIC yöntemleri için verilmiştir.

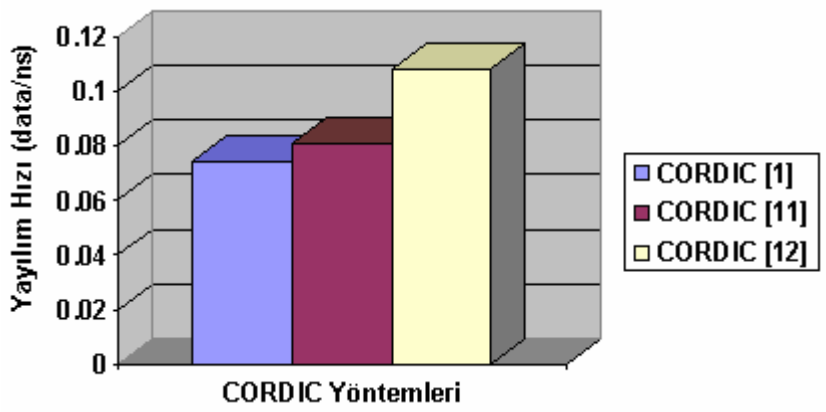
Analizi yapılan CORDIC yöntemleri iteratif çalışan mimarilerdir. Tablo 5-9 incelendiğinde [1]'in [11] ve [12]'ye göre FPGA donanımında daha fazla yazmaç, daha az dilim ve LUT kullandığı sonucuna ulaşılaşılmıştır.

Yöntemler	Yazmaç Sayısı	Dilim Sayısı	LUT Sayısı	Hata Oranı
CORDIC [1]	153	206	384	10^(-4) seviyelerinde
CORDIC [11]	116	537	1029	**Sinüs:** 4.97*10^(-5) **Kosinüs:** 2.02*10^(-5)
CORDIC [12]	85	433	827	**Sinüs:** 2.5*10^(-4) **Kosinüs**: 2.23*10^(-4)

Tablo 5 - 9. Farklı CORDIC yöntemleri için kaynak kullanımı ve hata oranlarının karşılaştırılması

Doğruluk performans kriteri açısından CORDIC yöntemleri karşılaştırıldığında [12]'nin, küçük bir çevrim tablosu kullanan [1] ile hemen hemen aynı doğrulukta sonuca ulaştığı gözlemlenmiştir. Bu durum Tablo 5-9'da gösterilmiştir. Koordinat sisteminin ilk quadrantı boyunca [12], (29-60] derece aralığında [11]'e göre yaklaşık 10 kat daha iyi doğrulukta sonuç vermektedir. Kartezyen koordinat sisteminin simetriği özelliğinden dolayı diğer quadrantlarda da aynı avantaj söz konusudur.

CORDIC yöntemleri hız açısından karşılaştırıldığında [12]'nin en yüksek yayılım hızına sahip olduğu sonucuna ulaşılmıştır. "Yayılım hızı" ifadesi ile "sinüs ve kosinüs giriş sinyallerinin değişmesinden, rotasyona uğramış sinüs ve kosinüs sonuç değerlerinin tanımlı ve kararlı olmasına kadar geçecek olan maksimal süre" belirtilmek istenmiş olup, Şekil 5-27'de farklı yöntemler için elde edilen yayılım hızlarının [data/ns] cinsinden dağılımı gösterilmiştir.

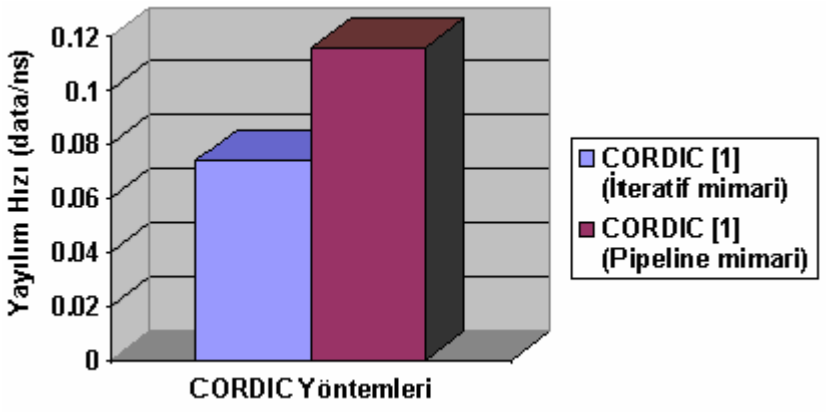


Şekil 5 - 27. Farklı CORDIC yöntemlerinin yayılım hızının karşılaştırılması

"Pipeline Çalışan Klasik CORDIC Algoritması" başlığı altında [1]'de açıklanan CORDIC algoritmasının pipeline olarak tasarım uygulaması verilmiştir. Pipeline çalışan CORDIC yöntemi için sentez sonuçları incelendiğinde donanımda kaynak kullanımının pipeline mimaride maksimum seviyeye çıktığı gözlenmiştir. Tablo 5 10'da bu durum gösterilmiştir. (İteratif yapıya göre 8.86 kat daha fazla alan kullanımı mevcut). Diğer bir yandan en yüksek yayılım hızına pipeline mimari ile ulaşılmıştır. Şekil 5-28'de Klasik CORDIC yönteminin [1] iteratif ve pipeline mimarilere sahip tasarımlarının yayılım hız sürelerinin [data/ns] cinsinden karşılaştırması verilmiştir.

Yöntemler	**Yazmaç Sayısı**	**Dilim Sayısı**	**LUT Sayısı**	**Hata Oranı**
CORDIC [1] (İteratif)	153	206	384	10^(-4) seviyelerinde
CORDIC [1] (Pipeline)	722	387	726	10^(-4) seviyelerinde

Tablo 5 - 10. Klasik CORDIC algoritması iteratif ve pipeline mimarilerinin donanımda kaynak kullanımı ve hata oranları açısından karşılaştırılması



Şekil 5 - 28. Klasik CORDIC algoritması iteratif ve pipeline mimarilerinin yayılım hızlarının karşılaştırılması

Benzer şekilde bu tez çalışmasında açıklanmış olan iteratif CORDIC yöntemleri de pipeline mimariye göre tasarlanmaya kalkışıldığında bir taraftan donanımda kaynak kullanımı artacak iken, diğer taraftan da yayılım hızı artacaktır. Pipeline uygulamalarda sonuca hızlı bir şekilde ulaşılabilmektedir.

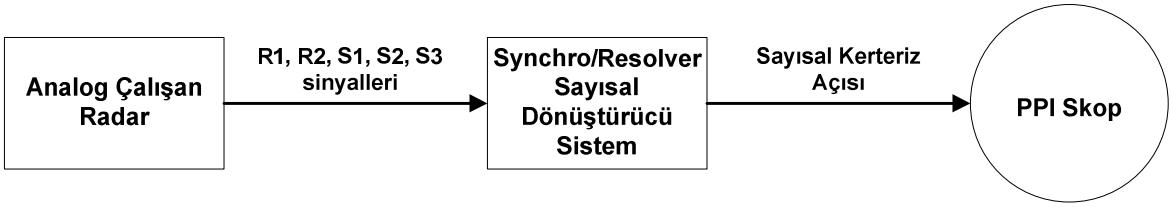
Gerçekleştirilen gömülü sistem uygulamalarında FPGA, trigonometrik fonksiyon hesaplaması görevi dışında başka görevler de üstlenmektedir. Bu bakımından donanımda optimum bir kaynak tüketiminin amaçlanarak tasarım yapılması önemlidir. Donanımda kaynak kullanımın artması kullanılacak olan FPGA'in dilim (slice) ve flip-flop sayısının da artacağı anlamına gelmektedir ki bu da tasarımcıya pahalı bir FPGA ürünü seçme zorunluluğu getirmektedir. Ayrıca donanımda kullanılan alanın artması güç tüketimini de artıracaktır. Gerçek zamanlı birçok gömülü sistem uygulamasında sistemin güç tüketimi kritik öneme sahiptir.

Trigonometrik fonksiyon hesaplamalarında doğruluk performans kriterinin kritik olduğu uygulamarda [1] veya [12] CORDIC yöntemleri kullanmak tasarımcıya avantaj getirebilir. Diğer bir yandan hem hız hem de doğruluk kritik öneme sahip ise [12]'nin tasarımcı için iyi bir çözüm olduğu söylenebilir. Sadece hız performans kriteri önemli ise [11]'i seçmek avantajlı olabilir.

6. Uygulama Alanlarına Detaylı Bakış

CORDIC algoritması çok sayıda problemin çözümünde kullanılmaktadır. Bu bölümde CORDIC algoritmasının kullanıldığı uygulamalardan birkaçı açıklanacaktır.

Takip eden sayfalarda “Synchro/Resolver Sayısal Dönüştürücü” uygulaması ve bu uygulamanın çıktılarını kullananan PPI skop uygulaması açıklanmış, bu uygulamalarda CORDIC algoritmasının ne amaçla kullanıldığı ifade edilmiştir. Bir sonraki uygulamada ise HFD katsatsayı üretiminde CORDIC algoritmasının kullanımının sağladığı avantajlar üzerinde durulmuştur.



Şekil 6- 1. Synchro/Resolver Sayısal Dönüştürücü-PPI skop uygulama arayüzü

6.1.1. Synchro Resolver – Sayısal Dönüştürücü Uygulaması

6.1.1.1. Ön Bilgi

“Synchro/Resolver'lar döner sistemdeki mil açısını elektriksel sinyallere çeviren aygıtlardır. Söz konusu bu aygıtlardan biri olan synchro'lar, mil ile sabit dönen rotor sargısının 120 derece açısal konumda bulunan yıldız bağlı üç adet stator sargısı arasındaki kuplaj oranına bağlı olarak konum bilgisini elektriksel olarak üretirler. Diğer bir aygıt olan resolver ise synchro'ya benzer biçimde çalışır. Resolver’ı synchro'dan ayıran tek fark stator sargılarının birbirlerine 90 derece açı ile konumlandırılmış olmasıdır.

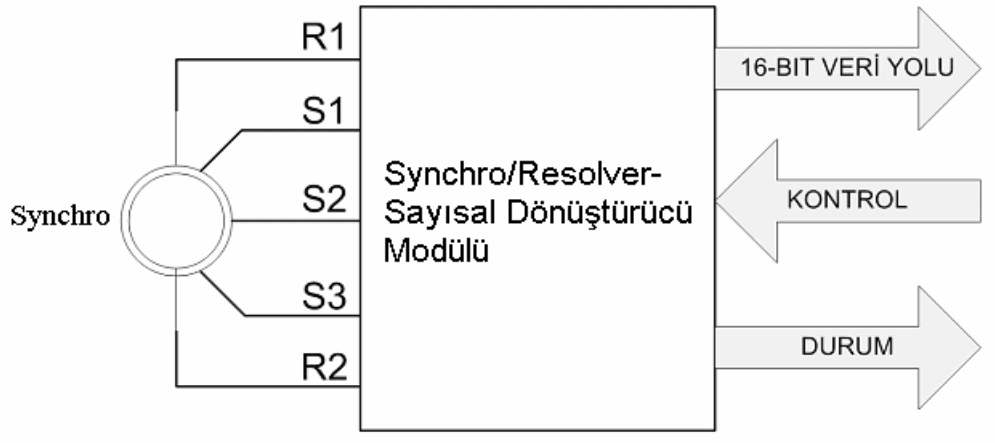
Sonuç olarak gerek synchro ve gerekse resolver, rotor'a indüklenmiş akımın stator sargılarının konuma bağlı kuple edilmesi ile konumun mekaniki sinüs ve kosinüs bileşenleri elektriksel analog verilere dönüştürülür. Bir çok radar sisteminde radar videosunun skop ekranında görüntülenebilmesi için “kerteriz açısı” bilgisine ihtiyaç duyulur. Söz konusu bu kerteriz açı bilgisi analog çalışan radarlar için doğrudan synchro/resolver'lardan elde edilmektedir. Radar videosunun sayısallaştırılarak

işlenebilir duruma getirilmesi için synchro/resolver sinyallerinin de sayısal bilgiye dönüştürülmesi gerekmektedir. Bu bağlamda “synchro/resolver-sayısal dönüştürücü”ler, analog synchro/resolver sinyallerinden sayısal açı değerlerini elde etmek için kullanılır.

Kerteriz açı bilgisinin yüksek çözünürlükte sayısallaştırılması, sayısallaştırılan video'nun da yüksek çözünürlükte işlenebilmesine imkan sağlamaktadır. “Synchro/resolver-sayısal dönüştürücü” uygulamaları için sayısallaştırma çözünürlüğü olarak 12-14-16 bit sıklıkla kullanılır. Böylece 360 derecelik tam bir tur kerteriz açısı 4096, 16384 veya 65536 parçaya bölümlenebilecektir. Kerteriz çözünürlüğünün artırılması ile ayrık zamanlı veri işleme (Discrete-Time Data Process) sonuçlarının da doğruluğu artırılmış olmaktadır.

Sonuç itibariyle “synchro/resolver-sayısal dönüştürücü” modüllerinde sayısallaştırılmış video işaretlerinin işlenebilmesi için ihtiyaç duyulan kerteriz bilgileri yüksek çözünürlükte elde edilebilmektedir.

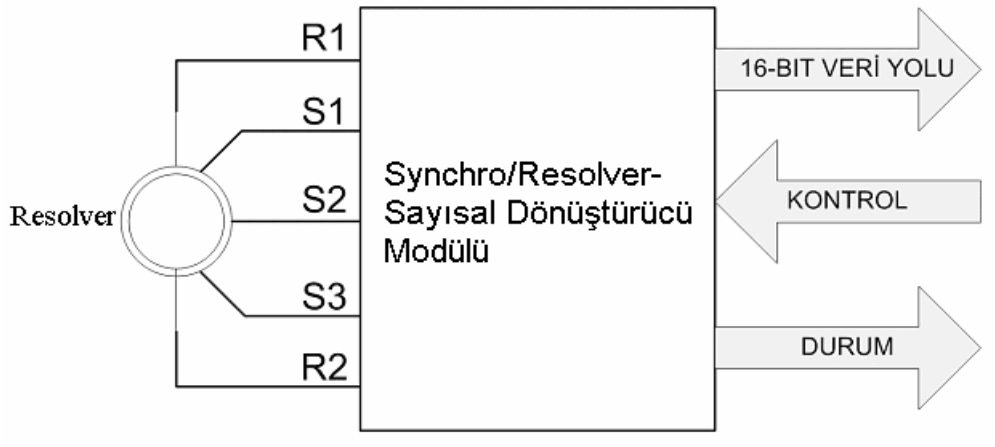
“Synchro/resolver-sayısal dönüştürücü” modülünün synchro ile bağlantısı Şekil 6-2’de gösterilmiştir.



Şekil 6- 2. Synchro Sistem Bağlantısı

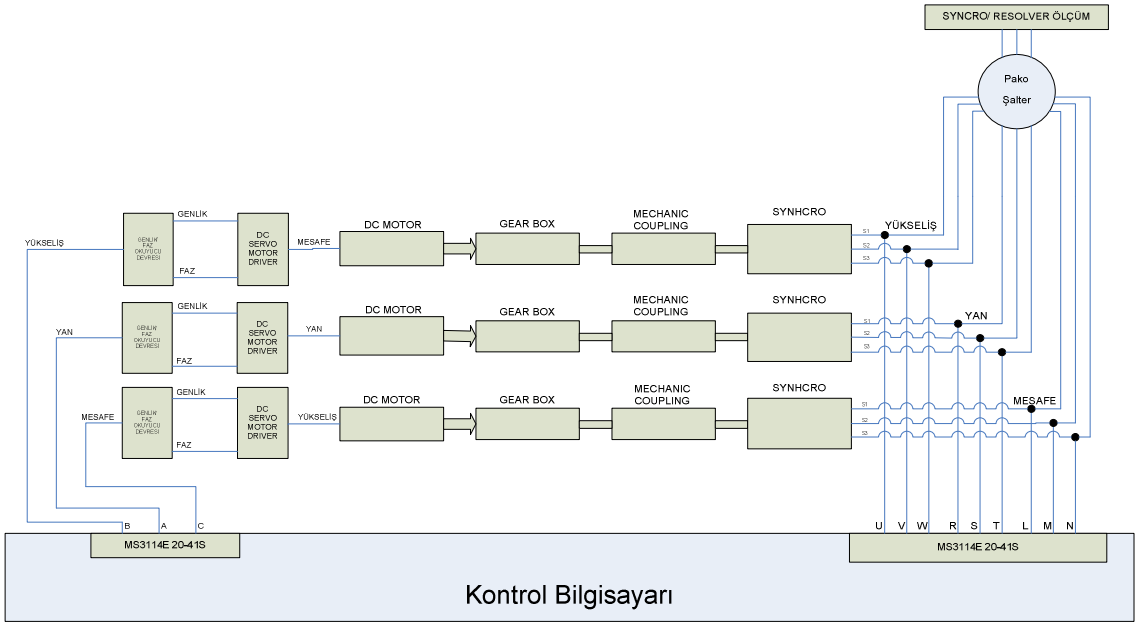
Şekilden de anlaşılacağı üzere “synchro/resolver-sayısal dönüştürücü” modülüne synchro'nun rotoruna uygulanan referans sinyalleri (R1 ve R2) ile birbirleri arasında 120 derece açıyla konumlandırılmış stator sargılarının ürettiği mil açısı bilgisini içeren elektriksel sinyaller (S1, S2 ve S3) giriş olarak bağlanmaktadır. Bu sinyaller modül içerisinde öncelikle Scott-T devresinden (mil açısının sinüs ve

kosinüs değerleri elde edilir) daha sonra analog-sayısal çeviricilerden geçerek sayısal bilgiye dönüştürülmekte ve dönüştürülen bu mil açısı bilgisi 16-bit veri yolu üzerinden modül çıkışından elde edilmektedir. “synchro/resolver-sayısal dönüştürü” modülünün resolver ile bağlantısı Şekil 6-3’de gösterilmiştir.



Şekil 6- 3. Resolver Sistem Bağlantısı

Yukarıdaki bağlantı şeklinin synchro bağlantısından farkı resolver'ın birbirleri arasında 90 derece açıyla konumlandırılmış stator sargılarına sahip olmasıdır. Böylece stator'da mil açısının sinüs ve kosinüs değerleri elektriksel sinyaller olarak üretilerek doğrudan sayısal dönüştürücü modüle aktarmış olmaktadır.” [S. Kabacalı, 9 Eylül 2009]. Tipik bir synchro-resolver sisteme ait genel mimari görünümü Şekil 6-4’de verilmiştir.

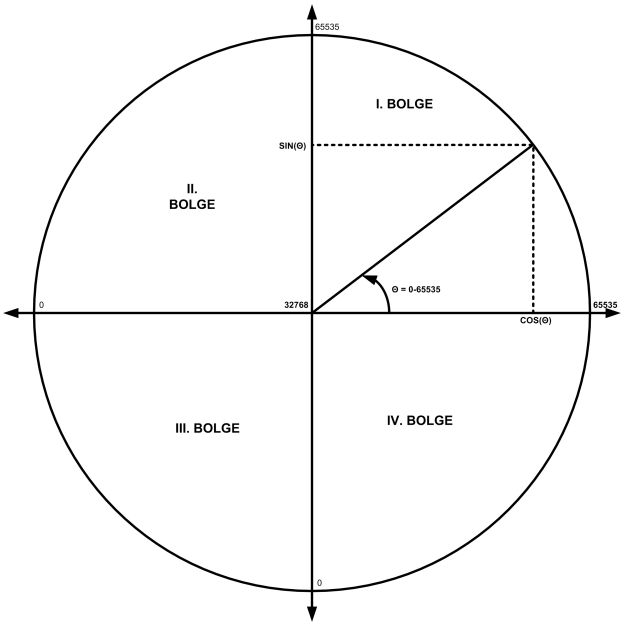


Şekil 6- 4. Synchro-resolver sistem mimarisi

6.1.1.2. Synchro Resolver Uygulamalarında CORDIC Algoritmasının Kullanımı

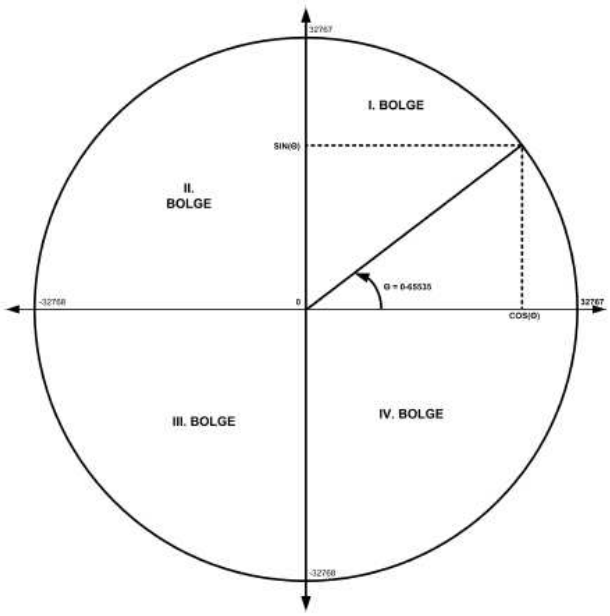
Bu tez çalışmasında dairesel CORDIC rotasyonları üzerinde yoğunlaşılmış olup, rotasyonel CORDIC modu temel alınmıştır. CORDIC algoritmasının uygulama alanlarına baktığımızda rotasyonel mod haricinde, vektörel mod temelli CORDIC rotasyonlarının da yaygın olarak kullanım alanı bulduğu gözlenmiştir. Bu bölümde rotasyonel ve vektörel modlarının her ikisini de içeren CORDIC yönteminin uygulamada nasıl kullanıldığı açıklanmıştır.

Sinüs ve kosinüs sonuçları ADC (Analog-to-Digital Converter) ile 16-bit üzerinden örneklendiği taktirde birim çembere ait nümerik değerler 0-65536 aralığında olacaktır.



Şekil 6- 5. Birim çembere ait nümerik değerler

16-bit ADC örnekleri $\sin\theta$ ve $\cos\theta$ eksenlerine yerleştirildiğinde birim çemberin orijin noktası (32768, 32768) olacaktır. Birim çemberin orijin noktasının (0,0) noktasına ayarlanması için gelen $\sin\theta$ ve $\cos\theta$ örneklerinden 32768 değerinin çıkarılması gerekmektedir. Orijin noktasının (0,0) 'a taşınması ile birim çemberin nümerik değerleri Şekil 6-6'deki gibi olacaktır. $\sin\theta$ ve $\cos\theta$ örneklerinin işaretleri, θ açısının koordinat uzayında hangi quadrantda bulunduğunu gösterir. Sınır değerler (0, 90, 180, 270 derece) için Tablo 6-1 esas alınarak işlem yapılır.



Şekil 6- 6. Birim çemberin orijin noktasının (0,0)'a ayarlanması

$\sin\theta$	$\cos\theta$	θ
0	32767	0
32767	0	16384
0	-32767	32768
-32768	0	49152

Tablo 6- 1. Sınır açı değerleri ve sinus-kosinüs karşılıkları

Sinüs ve kosinüs örnekleri kullanılarak elde edilecek θ açısının değer aralığının 0-65536 olduğu düşünülürse hedef açısının hangi değer aralığında ve koordinat uzayının hangi bölgesinde olduğununun tespiti Tablo 6-2'de gösterildiği gibi yapılmaktadır.

$\sin\theta$	$\cos\theta$	θ	θ Değer aralığı
+	+	1. Bölge (quadrant)	0-16383
+	-	2. Bölge (quadrant)	16384-32767
-	-	3. Bölge (quadrant)	32768-49151
-	+	4. Bölge (quadrant)	49152-65535

Tablo 6- 2. Koordinat uzayında bölge tespiti

Bölge tayini 16-bitlik θ açısı bilgisinin en önemli iki biti ile belirlenir. Kalan 14-bit θ açısının birinci bölgeye taşınması sonrasında elde edilir. Önceki bölümlerde açıklandığı üzere bu yöntem “alan katlama tekniği” (domain folding) olarak bilinmektedir. Bu amaçla giriş koordinat bileşenlerinin bulunduğu açısal bölgeye (quadrant) göre (6.1)-(6.6) denklemlerinden uygun olan çift işletilir.

$$\cos(\theta) = \sin(\frac{\pi}{2} + \theta) \quad II.Bolgeden\ I.Bolgeye \tag{6.1}$$

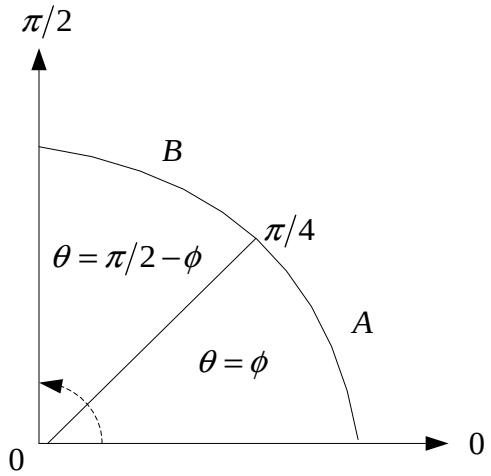
$$\sin(\theta) = -\cos(\frac{\pi}{2} + \theta) \quad II.Bolgeden\ I.Bolgeye \tag{6.2}$$

$$\sin(\theta) = -\sin(\pi + \theta) \quad III.Bolgeden\ I.Bolgeye \tag{6.3}$$

$$\cos(\theta) = -\cos(\pi + \theta) \quad III.Bolgeden\ I.Bolgeye \tag{6.4}$$

$$\cos(\theta) = -\sin(\frac{3\pi}{2} + \theta) \quad IV.Bolgeden\ I.Bolgeye \tag{6.5}$$

$$\sin(\theta) = \cos(\frac{3\pi}{2} + \theta) \quad IV.Bolgeden\ I.Bolgeye \tag{6.6}$$



Şekil 6- 7. İlk quadrantın iki alana parçalanması ve hedef açısının belirlenmesi

Alan katlama tekniği kullanılarak θ açısı birinci bölgeye alındıktan sonra birbirlerini 90 dereceye tamamlayan açıların da sinüs ve kosinüs değerlerinin birbirine eşit olacağı göz önüne alınarak ilk quadrant iki alana bölünür.

Örneklenen $\sin\theta$ değeri sin45 değerinden (23170) küçük ise $\sin\theta$ değeri esas alınır, örneklenen $\sin\theta$ değeri sin45 değerinden büyük ise $\sin\theta$ ve $\cos\theta$ değerleri yer değiştirilir. Bu durumda hesaplanan θ açısından gerçek açıya gitmek için bulunan açı değeri 16384'den çıkarılır. Bu durum Şekil 6-8'deki sözde kod ile ifade edilmiştir.

```
if  (x ≥ y) then            // if  (y < 23170)  Domain A
        x' = x
        y' = y
        θ = θ
else                        Domain B
        x' = y
        y' = x
        θ = 16384 – θ       // θ = (π/2 – θ)
endif
```

Şekil 6- 8. Alan tespiti

θ açısının hesaplanabilmesi için 1. Bölgeye ait açılar tablolanır. Bu durum Tablo 6-3'de gösterilmiştir.

i	θ Değer Aralığı (0-8192)	$\sin\theta$ Değer Aralığı (0-32767)	$\cos\theta$ Değer Aralığı (0-32767)
0	8192	23170	23170
1	4096	12540	30274
2	2048	6393	32138
3	1024	3212	32610
4	512	1608	32729
5	256	804	32758
6	128	402	32766
7	64	201	32767
8	32	101	32767
9	16	50	32767
10	8	25	32767
11	4	13	32767
12	2	6	32767
13	1	3	32767

Tablo 6- 3. Açı bilgilerinin tabloda tutulması

$\sin\theta$ ve $\cos\theta$ örneklerinin tablo ile karşılaştırılması yapılarak $\sin\theta$'nın hangi değerden küçük, $\cos\theta$'nın hangi değerden büyük olduğu tespit edilir.

Sinüs değeri alınan örneğin $\sin\theta$ değerinden küçük olduğu sürece tabloda aşağı yönde ilerlenir ve yeni sinüs, kosinüs değerleri Denklem (2.11) kullanılarak hesaplanır.

Herhangi bir anda tablodan bulunan sinüs değeri $\sin\theta$ değerinden büyük olursa tabloda ilgili adım algoritmada işletilmez. Yukarıda anlatılanları pekiştirmek amacıyla aşağıda bir örnek verilmiştir.

ÖRNEK-10:

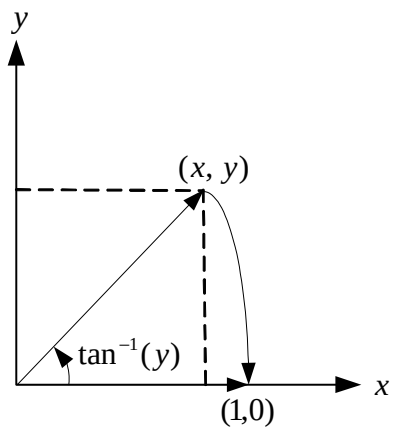
Synchro/resolver sistemde stator'da mil açısının sinüs ve kosinüs değerleri elektriksel sinyal olarak üretilerek doğrudan sayısal dönüştürücü modüle aktarıldığında, sayısal dönüştürücü modülünün çıkışındaki 16-bitlik veri yolundan okunan $\sin\theta$ ve $\cos\theta$ değerlerinin sırasıyla 59334 ve 13585 olduğunu varsayalım. Bu durumda konum bilgisini verecek olan θ hedef açısı (radar video sayısallaştırma uygulamalarında kerteriz açısı) değerini hesaplayalım.

Vektörel modu girişleri: Başlangıç vektörü koordinat bileşenleri

$\sin\theta$ = 59334

$\cos\theta$ =13585

Vektörel modu çıkışlar: Vektör uzunluğu ve açı



Şekil 6- 9. Vektörel modu vektör rotasyonu

Birim çemberden yararlanılarak vektörel modundaki işlem adımları aşağıdaki gibi açıklanabilir.

Vektörel Mod İşlem Adımları:

$x_0 = x$, $y_0 = y$ başlangıç değerleri için:

- (x, y) başlangıç vektör koordinatları için başlangıç vektörünü y=0 olana kadar döndür. İteratif işlem adımları boyunca y değeri sıfıra yakınsat. ($y_n \cong 0$)
- Vektörün dönüş açısını hesapla.
- Dönüş açısı = $\arctan(y)$ atamasını yap.

Rotasyonel modu girişleri: Başlangıç vektörü koordinat bileşenleri, rotasyon açısı

Rotasyonel modu çıkışlar: Yeni vektörün koordinat bileşenleri

Aşağıda adım adım açıklanacak olan CORDIC algoritması her iki rotasyon modunu da kullanmaktadır.

Adım-1: Orijin noktasını (0,0)'a taşıyın.

$$\sin\theta = 59334 - 32768 = 26566$$

$$\cos\theta = 15385 - 32768 = -19183$$

Adım-2: Açının kartezyen koordinat düzleminde hangi bölgede (quadrant) olduğunu tespit edin.

Tablo 6-2'ye göre θ hedef açısı 2. bölgededir.

Adım-3: Bulunan bölgeyi, birinci bölgeye taşıyın.

Koordinat uzayının birinci bölgesi $(0, \pi/2)$ aralığı 16384 açısal değerine karşılık geldiğinde göre, alan katlama tekniğini kullanarak, giriş sinüs ve kosinüs bileşenlerini ilk bölgeye taşıyabilmek için (6.1) ve (6.2) nolu denklemler kullanılır.

Bu durumda;

$\sin\theta = 19183$

$\cos\theta = 26566$

değerlerine dönüşür.

Adım-4: θ açısının 45 dereceden büyük veya küçük olduğunu test edin.

Şekil 6-13 göz önüne alındığında $\sin\theta$ değeri 23170'den küçük olduğu için θ açısı 45 dereceden küçüktür. Bu sebeple $\sin\theta$ ve $\cos\theta$ değerleri yer değiştirmez.

Adım-5: $\sin\theta$'dan küçük olan sinüs değerini tablodan bulun. $\sin\theta$ değerinden küçük olan ilk değer $\sin(\alpha_2) = 12540$ ve $\cos(\alpha_2) = 30274$ 'dir. O halde $\alpha_2 = 4096$ olacaktır. Bu şekilde hesaplanan sinüs değeri, $\sin\theta$ değerinden küçük olduğu sürece algoritmayı (2.11) nolu denklemine göre iteratif bir şekilde yürütecek olursak Tablo 6-4'de verilen sonuçlar elde edilir.

i	θ Değer Aralığı (0-16383)	$\sin\theta$ Değer Aralığı (0-32767)	$\cos\theta$ Değer Aralığı (0-32767)	Algoritma işletilsin mi?	$\sin(\alpha_i)$	$\cos(\alpha_i)$
0	8192	23170	23170	hayır		
1	4096	12540	30274	evet	12540	30274
2	2048	6393	32138	evet	18204	27246
3	1024	3212	32610	hayır	20787	25330
4	512	1608	32729	hayır	19519	26320
5	256	804	32758	evet	18867	26791
6	128	402	32766	hayır	19195	26558
7	64	201	32767	evet	19031	26674
8	32	101	32767	evet	19113	26615
9	16	50	32767	evet	19153	26585
10	8	25	32767	evet	19173	26570
11	4	13	32767	evet	19183	26561
12	2	6	32767	hayır	19187	26556
13	1	3	32767	hayır	19185	26558

Tablo 6- 4. İteratif algoritmanın yürütülmesi

Çıkış verileri:

Yürütülen mikro rotasyon açıları: $\alpha_i = [0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0]$

$$\sin(\alpha) = 19183 = 19183 * 2^{-15} = 0.5854$$

$$\cos(\alpha) = 26561 = 26561 * 2^{-15} = 0.8106$$

$$\sum_{i=0}^{13} \theta_i = 4096 + 2048 + 256 + 64 + 32 + 16 + 8 + 4 = 6524$$

Giriş olarak verilen $\sin\theta$ ve $\cos\theta$ değerleri koordinat uzayının 2. bölgesinde bulunduğu için açı değerlerinin toplam yakınsama değerine bir bölgelik açısal fark değeri olan 16384 eklenir.

O halde;

$\theta = 16384 + \sum_{i=0}^{13} \theta_i = 22908 = 22908$ elde edilir.

Vektörel mod işlemler burada devreye girer.

Vektörel mod dönüş açısı arktanjant değeri algoritma seviyesinde hesaplanarak

$\theta = 35.84^0$ değeri elde edilir. $\theta = \cos'(0.8106) = \sin'(0.5854)$

Sonuçların yorumlanması:

sinüs bileşeni hatası: 19183-19183=0

kosinüs bileşeni hatası: 26566-26561=5

açı değeri hatası: 22907-22908= -1 (%0.0055 hata oranı mevcut)

olarak gözlenmiştir.

Yukarıdaki açısal hata değerine neden olan faktör kayan nokta işlemler yerine tam sayı işlem yapılmış olmasıdır.

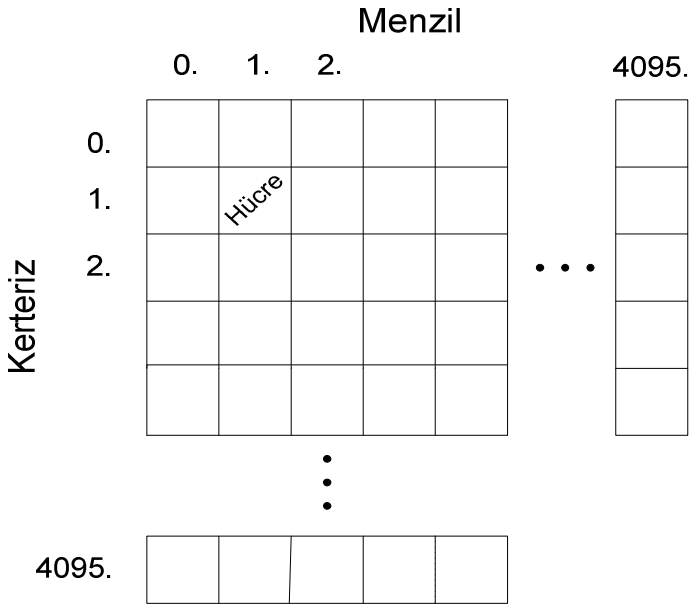
6.1.1.3. PPI Skop Grafik Arayüz Uygulamalarında CORDIC Algoritmasının Kullanımı

Analog çalışan radarlarda synchro ve resolver rotor'a indüklenmiş akımın stator sargılarının konuma bağlı kuple edilmesi ile hedef konumunun mekaniki sinüs ve kosinüs bileşenleri elektriksel analog verilere dönüştürülür. Bir çok radar sisteminde radar videosunun skop ekranında görüntülenebilmesi için "kerteriz açısı" (azimuth) bilgisine ihtiyaç duyulur. Söz konusu bu kerteriz açı bilgisi analog çalışan radarlar için doğrudan synchro/resolver'lardan elde edilmektedir. Şekil 6-10'da PPI skop ve PPI skop'a sayısal kerteriz açı bilgisini gönderen Synchro/Resolver Sayısal Dönüştürücü sistem gösterilmiştir.

Şekil 6- 10. PPI skop arayüzü

6.1.2. PPI Ekranda Sinüs-Kosinüs Koordinat Hesabı

SRVÖ (Sayısallaştırılmış Radar Video Örnekleri) anolog radar videosunun donanımsal bir çözüm ile sayısallaştırılmış halidir. SRVÖ video işaretlerinin enerji seviyelerini temsil etmektedir. Radarın bir turu için SRVÖ 4096x4096 boyutunda bir matrise karşılık gelmektedir. Matrisin boyutu, radar tipine ve sistem ayarlarına göre değişebilmektedir. Bu matrisin her bir elemanı "hücre" olarak isimlendirilmektedir. Her hücre, belirli bir kerterizdeki menzil değerini temsil etmektedir.



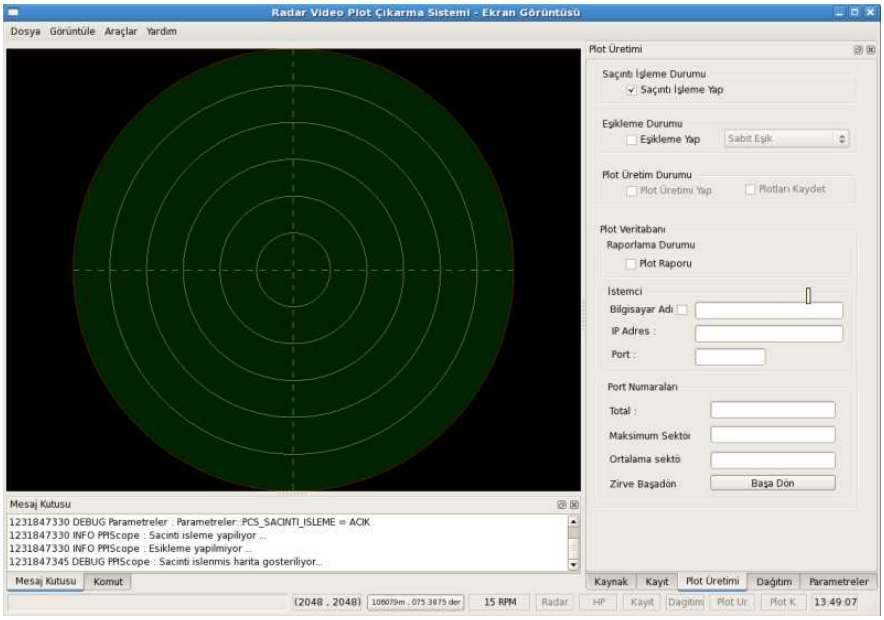
Şekil 6- 11. Kerteriz-Menzil Haritası

PPI ekran modülü plot çıkartma uygulama yazılımından gelen radar video görüntüsünün kullanıcı arayüzünü oluşturur. Sayısallaştırılmış radar videosu, saçıntı işleme algoritması sonucu çıkmış video (clutter processed), seviyelendirilmiş video (thresholded) ve saçıntı haritası (clutter map) seçmeli olarak bu ekranda gösterilir.

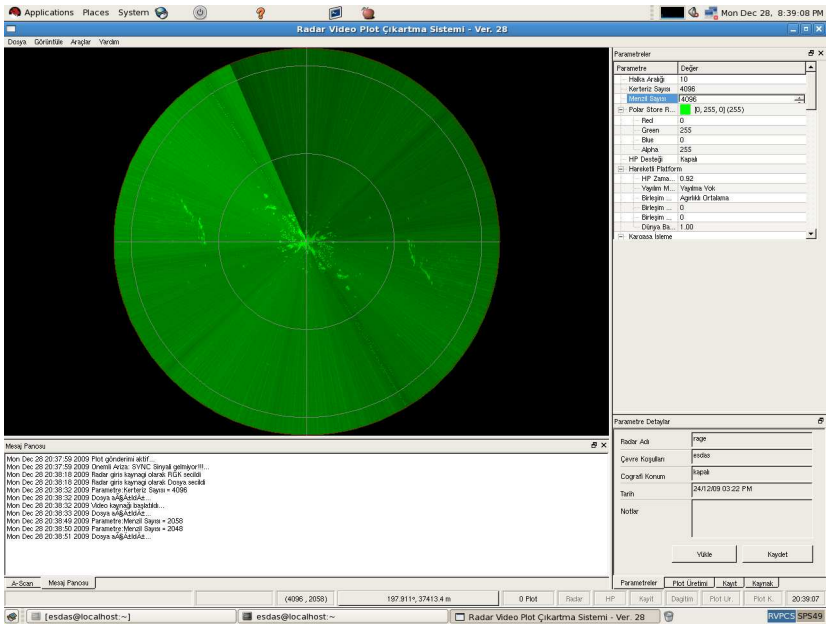
Sayısallaştırılmış video görüntüsünde hedeflere ait kerteriz ve menzil konum bilgisi synchro/resolver sayısal dönüştürücü biriminden PPI skop'a aktarılır. PPI skop gelen kerteriz ve menzil değerlerinin karşılık geldiği sinüs ve kosinüs bileşenlerini hesap eder ve polar koordinat sisteminde uygun noktalara yerleştirir. Bu işlem için polar koordinat sisteminde mümkün olabilecek tüm polar noktalara ait sinüs ve

kosinüs bileşenlerinin önceden hesaplanmış ve bir tabloda tutulması öngörülebilir veya algoritmik hesaplamalar sırasında sinüs ve kosinüs değerleri anlık olarak CORDIC algoritması kullanılarak hesaplanabilir. PPI ekranda gösterilen radar video işareti, tanımlı eşik seviyesi üzerinde ise plot olarak kabul görür ve mevcut radar işareti bir dörtgen içersine alınır. Bu dörtgene “plot” adı verilmekte olup, plot çıkartma operasyonlarında da sinüs ve kosinüs bileşenlerinin hesaplanmasına ihtiyaç duyulur. Bu noktada yine CORDIC algoritmaları devreye girer.

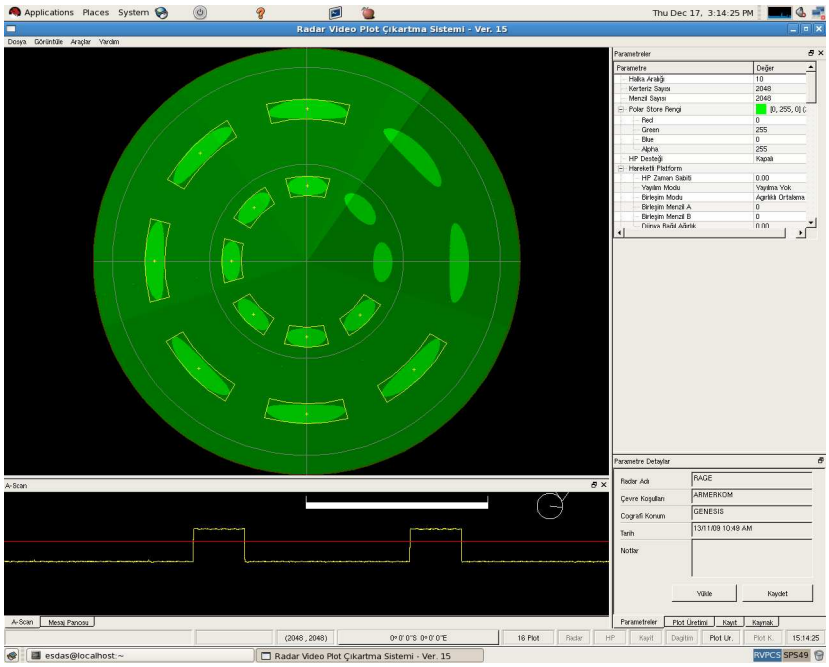
Şekil 6-12, 6-13, 6-14 ve 6-15’de PPI ekran grafik kullanıcı arayüzü görüntüleri verilmiştir.



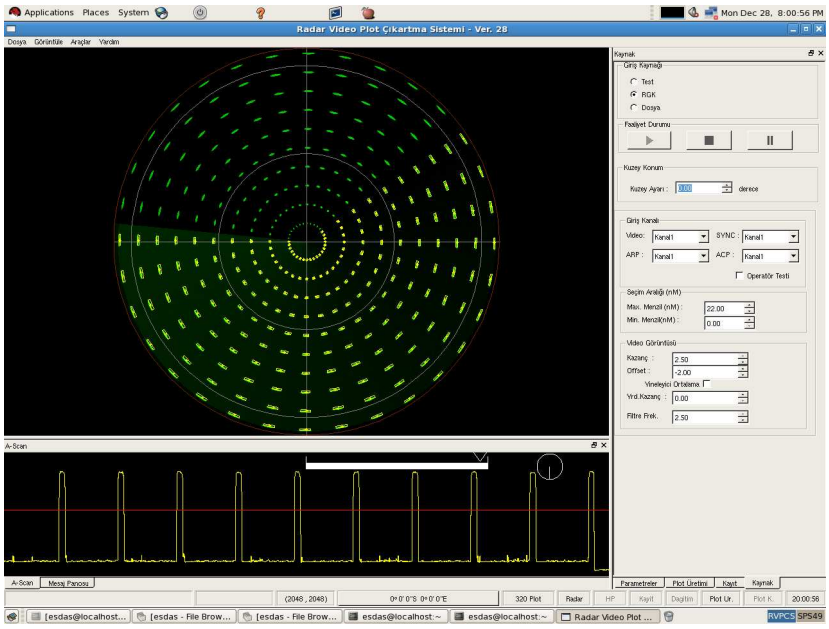
Şekil 6- 12. PPI ekrana sahip radar video plot çıkartma sistemi grafik kullanıcı arayüzü



Şekil 6- 13. Radar PPI ekranında karaya konuşlu SPS-10 radarına ait görüntü



Şekil 6- 14. Radar PPI ekranında 16 hedefin bulunduğu bir test paterninden plot üretilmesi



Şekil 6- 15. Radar PPI ekranında 320 hedefin bulunduğu bir test paterninden plot üretilmesi

6.1.3. Hızlı Fourier Dönüşümü

6.1.3.1. Ön Bilgi

HFD modern sayısal sinyal işleme sistemlerinde kullanılan en popüler DSP hesaplama algoritmasıdır.

Bu algoritmayı açıklayacak olursak;

"Ayrık-Fourier dönüşümünün doğrudan hesaplanmasında her bir X_k değeri için N karmaşık çarpma ve $N-1$ karmaşık toplama işlemi kullanılmaktadır. Bu nedenle N adet AFD (Ayrık Fourier Dönüşümü) değeri bulunurken, N^2 çarpma ve $N.(N-1)$ toplama işlemi gereklidir.

Ayrıca, her karmaşık çarpma işlemi 4 gerçel çarpma, 2 gerçel toplama işlemi ve herbir karmaşık toplama 2 gerçel toplama işlemi ile gerçeklenmektedir. Sonuç olarak, dizi uzunluğu olan N'nin büyük olması durumunda doğrudan AFD bulunması çok fazla miktarda işlem gerektirmektedir. Yani, N sayısı artarken gereken işlem sayısı yüksek hızla artmaktadır.

AFD hesaplamalarında etkin ve günümüzde kullanılan yaklaşım, Hızlı Fourier dönüşümü (HFD) algoritmalarıdır. HFD, AFD'den farklı bir algoritma değildir. HFD, AFD hesaplanması için etkili, ekonomik bir algoritmadır.

AFD'un sayısal işaret işleme alanında spektrum analizi, konvolüsyon ve korelasyon gibi işlemlerin gerçeklenmesinde önemli rol oynaması daha etkin HFD algoritmalarının ortaya çıkmasına yol açmıştır." [A. Hamdi Kayhan, 2004]

6.1.3.2. HFD Katsayısı Üretiminde CORDIC Uygulamasının Avantajları

Bu uygulamalarda güç tüketimi ve donanımda kullanılan alan birincil derecede öneme sahip performans kriterleridir. Güç ve alan verimli uygulamalar için HFD işlemlerinin donanımda harcadığı alanı miminimize etmek gerekmektedir.

Bu amaçla tasarımcılar, tabloda tutulan HFD katsayılarının kapladığı hafıza alanının minimuma getirilmesine ihtiyaç duymuşlardır. Bu noktada devreye CORDIC algoritmaları girmektedir. Önceden hesaplanmış HFD katsayılarının bir tabloda tutmak yerine CORDIC algoritması kullanımı ile HFD katsayılarını çarpma operasyonuna ihtiyaç kalmadan üretmek mümkündür. Bu şekilde HFD hesaplamalarındaki yüksek hesaplama karmaşıklığı düşürülmüş olmakta, donanımda kullanılan alan ve güç tüketim değerleri düşürülmektedir. Bu bakımdan CORDIC algoritması, HFD algoritmasını daha yüksek performanslarda kullanma imkanı verir. Sonuç itibariyle, FFT vb. uygulamalarında yüksek doğruluk ve hızda sonuca ulaşılmak istenen uygulamalarda CORDIC algoritmaları kullanımı tercih edilir.

7. SONUÇLAR

Yoğun çarpma işlemi gerektiren donanım tabanlı uygulamalarda istenen yüksek hız, donanımda az yer tüketme, düşük güç tüketimi, düşük maliyet gibi önemli performans isterlerine karşılık verebilmek için tasarımcılar yapısında donanım çarpıcıları bulunduran yüksek maliyetli programlanabilir cihazlar yerine yapısında donanım çarpıcısına sahip olmayan programlanabilir cihazlarda algoritma seviyesinde çözüm sağlamayı isterler. Bu yapılarda tasarımcılar donanım çarpıcısı çözümü yerine donanımda küçük çevrim tablosu kullanan CORDIC algoritmaları veya donanımda çevrim tablosu kullanmayan CORDIC algoritmalarını kullanmayı tercih etmektedirler.

Çevrim tablosu uygulamaları aynı sonucu veren donanımsal uygulamalar ile harcanan işlem zamanı bakımından karşılaştırıldığında, ROM uygulamalarının en hızlı hesaplama yöntemlerinden biri olduğu söylenebilir. Ancak ROM kullanımı donanımda bellek kullanımı ihtiyacını doğurmaktadır ve ROM uygulamalarında sonucun doğruluğu, ROM'un boyutu ile sınırlandırılmıştır. Yüksek doğrulukta sonuçlara ulaşılmak istendiği durumlarda donanımda kullanılan uçucu olmayan bellek alanının da artacağı açıktır. Gömülü sistem uygulamalarında özellikle de radar vb.. donanımda kullanılan alanın kritik olduğu çalışmalarda, ROM kullanımı ilave adres/veri yolunu gerektireceğinden bu yöntemi kullanmak beraberinde dezavantajları da getirmektedir. Trigonometrik fonksiyon hesaplamaları genellikle sinyal işleme vb. giriş-çıkış uygulamalarında sıkça kullanıldığından bu uygulamalarda genellikle programlanabilir bir aygıt bulunmaktadır. Programlanabilir bir aygıt, görevlerinden biri olan trigonometrik fonksiyon hesaplamasını yaparken çevre birimi olarak sinüs-kosinüs değerlerinin tutulduğu ROM kullanmak yerine tüm operasyonları kendi yapısı içersinde çözmesi tasarım açısından istenilen bir durumdur. Bu tarz mimarilerde yüksek doğrulukta sonuçlara ulaşılabilmek için ROM boyutunun artırılması, hem tasarlanan kartın boyutunu (adres yolu büyüdüğü için ve ROM kullanıldığı için) hem de hem de tasarım maliyetini artıracaktır.

Donanım çarpıcılarının bulunmadığı programlanabilir aygıt uygulamalarında ise kullanılabilecek en hızlı trigonemetrik hesaplama yöntemi CORDIC'dir. CORDIC algoritmaları bu tez çalışmasının temelini oluşturmaktadır.

Tez çalışması kapsamında geliştirilen “GUICORDIC” MATLAB yazılımı, farklı mimarilere sahip sinüs ve kosinüs trigonometrik fonksiyonlarının algoritmik simülasyonunu; iterasyon sayısı, sonuç değerleri ve doğruluk kriterleri açısından analiz edebilen bir yeteneğe sahiptir. Bu yazılım sayesinde GUICORDIC grafik kullanıcı arayüzünü kullanan bir kullanıcı bilgi ekranları yardımıyla algoritmalar arasında performans karşılaştırmasını kolaylıkla yapabilmektedir.

MATLAB yazılımının tasarlanmasındaki en önemli amaç; sinüs ve kosinüs hesaplamalarını yapan klasik yöntemlerle, çevrim tablosu kullanmayan CORDIC yöntemlerinin yukarıda sayılan performans kriterleri açısından simülasyon ortamındaki etkilerini inceleyebilmektir. MATLAB yazılımı altı adet sinüs-kosinüs hesaplama yönteminden oluşmaktadır.

Bu yöntemlerden ikisi klasik sinüs-kosinüs hesaplama yöntemleri olan çevrim tablosu temelli yöntem ve Taylor serisi kullanan yöntemdir. Diğer bir sinüs-kosinüs hesaplama yöntemi, trigonometrik fonksiyon hesaplamalarında bir dönüm noktası olan küçük bir çevrim tablosu kullanan CORDIC algoritmasıdır. Kalan diğer üç yöntem ise bu tez çalışmasının temelini oluşturan hesaplamalarda çevrim tablosu kullanımını tamamiyle ortadan kaldıran etkin CORDIC algoritmalarıdır. MATLAB yazılımı sinüs-kosinüs hesaplama yöntemlerinin gelişim sırasına göre önemli görülen mimarileri göz önüne alarak tasarlanmış olup, kullanıcı istediği hesaplama yöntemini seçerek hesaplama ve performans analizleri yapabilmektedir.

Bu tez çalışmasının temelini oluşturan bir diğer önemli unsur çevrim tablosu kullanmayan CORDIC algoritmalarının donanımsal uygulamasının FPGA ile gerçekleştirilmesidir. Bu amaçla VHDL yazılımı, Xilinx simülasyon ve sentezleme araçları kullanılarak, MATLAB ortamında geliştirilen algoritmalar, donanım seviyesinde tasarlanmıştırlar. VHDL yazılımı kullanılarak yazılan tümdevre tasarım kodları RTL (Register Transfer Seviyesi) sentezleme ile kapı seviyesi netlist formuna dönüştürülür. Sentezleme sonucunda FPGA yongasının ne kadarının kullanıldığı ve bunların istatistiksel analizi beşinci bölümde yapılmıştır. Ayrıca aynı bölüm içersinde Xilinx ISE programında test kodları (testbench) oluşturularak algoritmaların Xilinx ISE programında simülasyonu yapılarak, donanımsal performans karşılaştırmaları gerçekleştirilmiştir.

MATLAB simülasyonu ve FPGA uygulama sonuçlarına göre trigonometrik hesaplama yöntemleri karşılaştırıldığında çevrim tablosu kullanmayan CORDIC algoritmalarının hızlı ve donanımı etkin kullandıkları ortaya çıkmıştır. Ayrıca bu çalışmada adı geçen “Highly Accurate Reduced Iterations CORDIC Processor Algorithm” [12] yayını bu tez çalışmasının bir ürünüdür. [12] ile küçük bir çevrim tablosu kullanan Klasik CORDIC [1] yaklaşımına çok yakın doğruluk seviyesinde ve düşük iterasyonda sonuçlara ulaşmak mümkündür. Önerilen CORDIC algoritması [12], [11]’de açıklanan CORDIC algoritması ile karşılaştırıldığında, maksimum 10 iterasyon için %8.59 , maksimum 16 iterasyon için ise % 5.39 daha düşük iterasyonda ve daha yüksek doğruluk oranlarında sonuçlar elde edilmiştir. Hız, donamımda kullanılan alan, doğruluk, yayılım hız performans kriterleri göz önüne alındığında uygulama amacına göre [12]’nin kullanımının avantaj getireceği sonucuna ulaşılmış olup, uygulamadaki öncü performans kriterine göre donanım çarpıcısı olmayan sistemlerde, incelenen CORDIC algoritmalarından istenilen özellikte olanı olumlu sonuçlar verecektir.

Tez boyunca açıklanan tüm CORDIC algoritmaları dairesel CORDIC rotasyonları ile sınırlı tutulmuştur. Sinüs ve kosinüs fonksiyonlarının kombinasyonları ile birçok trigonometrik fonksiyon hesaplaması kolaylıkla yapılabilmektedir.

8. ÖNERİLER ve TARTIŞMA

Gelişen FPGA teknolojisinde azalan risk, düşük geliştirme zamanı ve tasarım esnekliği gibi faktörlerden dolayı IP (Fikri Mülkiyet) çekirdek tasarımları yaygınlaşmıştır. Bu tasarımlar donanımsal veya yazılımsal olabilir.

Donanımsal IP çekirdeği mantık tasarımında kullanılan, önceden test edilmiş karmaşık seviyedeki işlevlerdir. IP çekirdeği kullanımı ile donanım geliştirme sürecinde basitlik ve zaman tasarrufu, doğrulama zamanının azalması ve verimin artması sağlanır. Donanımsal IP önceden gerçeklenmiş çok karmaşık yapıda bir mikro işlemci çekirdeği, giga-bit alıcı verici, çarpıcı, toplayıcı, MAC birimi gibi öbekleri kapsar. Bu öbekler güç tüketimi, alan kullanımı ve başarım açılarından daha verimli olmayı sağlar. Yazılımsal IP, yüksek seviyeli işlevler için hazırlanmış kaynak kütüphanelerinin kullanıcı tarafından tasarıma eklenmesini sağlar. Bu tasarımlar karmaşık seviyeli işlevler içerir. Bu tez çalışmasının bir sonraki adımı geliştirilmiş çevrim tablosu kullanmayan CORDIC algoritmalarının IP çekirdeklerinin spesifikasyonlarının belirlenerek tasarlanması olabilir.

Yaygın bir inanışa göre DSP mikroişlemcilerinin, CORDIC algoritması performansını iyileştiremedikleri söylenmektedir. İlk bakışta bu durum makul görülebilir, çünkü CORDIC algoritması donanım çarpıcısı bulunmayan donanım mimarilerinde oldukça etkindir. Donanım çarpıcısı bulunan sistemlerde CORDIC algoritmasının performansının iyileştirilip iyileştirilemeyeceği günümüzde merak edilen araştırma konularından birisidir. CORDIC rotasyonel denklemlerinin yeniden düzenlenmesi ile DSP mikroişlemci temelli donanım mimarilerinde CORDIC kullanımı avantaj sağlayabilir. [15]'de bu konu üzerinde durulmuştur.

KAYNAKLAR DİZİNİ

[1] Volder, J. E., 'The trigonometric computing Technique', *IRE Transactions on Electronic Computing*, 1959, pp. 330-334.

[2] Walther, J. S., 'A unified algorithm for elementary functions', *in AFIPS Spring joint Computer Conference*, 1971, vol.38. pp. 379-385.

[3] Despain, A.M., 'Fourier Transform Computers Using CORDIC Iterations', *IEEE Transactions on Computers*, 1981, vol. 30, no. 10, pp. 993-1001.

[4] Banerjee A., Dhar, A.S. and Banerjee, S., 'FPGA realization of a CORDIC Based FFT processor for biomedical signal processing', *Microprocessor & Microsystems*, May, 2001, vol 25/3, pp. 131-142.

[5] Hemkumar, N.D. and Cavallaro, J.R., 'Efficient complex matrix transformations with CORDIC', *in Proc. 11th Symposium on Computer arithmetic*, Windsor, ON, Canada, June 29-July 2, 1993, pp. 122-129.

[6] Cavallaro, J. R. and Luk, F.T., 'CORDIC Arithmetic for an SVD Processor', *Journal of Parallel and Distributed Computing*, June 1988, Volume 5, No. 3, pp. 271-290.

[7] Euh, J., Chittamuru, J. and Burleson, W., **'**CORDIC vector interpolator for power-aware 3D computer graphics', *IEEE Workshop on Signal Processing Systems, 2002, pp. 426-431.*

[8] Hsiao, S.F. and Delosme, J.M. 'The CORDIC householder algorithm', *Proc. 10th Symposium on Computer arithmetic,* 1991, pp. 256-263.

[9] Maharatna, K., Dhar, A.S., and Banerjee, S.: 'A VLSI array architecture for realization of DFT, DHT, DCT and DST', Signal Process., 2001, 81, pp. 1813–1822.

[10] Koushik Maharatna, S. Banerjee, E. Grass, M. Krstic, A. Trayo, "Virtually Scaling Free Adaptive CORDIC Rotator Algorithm and Architecture," IEEE Trans. On Circuits and Systems for Video Techn. VOL. 15, NO.11, NOVEMBER 2005.

[11] R. Zhang, J. Hun Han, A. T. Erdoğan, T. Arslan, “Low Power CORDIC IP Core Implementation,” ICASSP 2006, IEEE 2006.

[12] E. Aytöre, A. Z. Alkar, “Highly Accurate Reduced Iteration CORDIC Processor Algorithm,” International Journal of Electronics, Taylor&Francis Group, Vol. 97, No. 2, February 2010, 163–176.

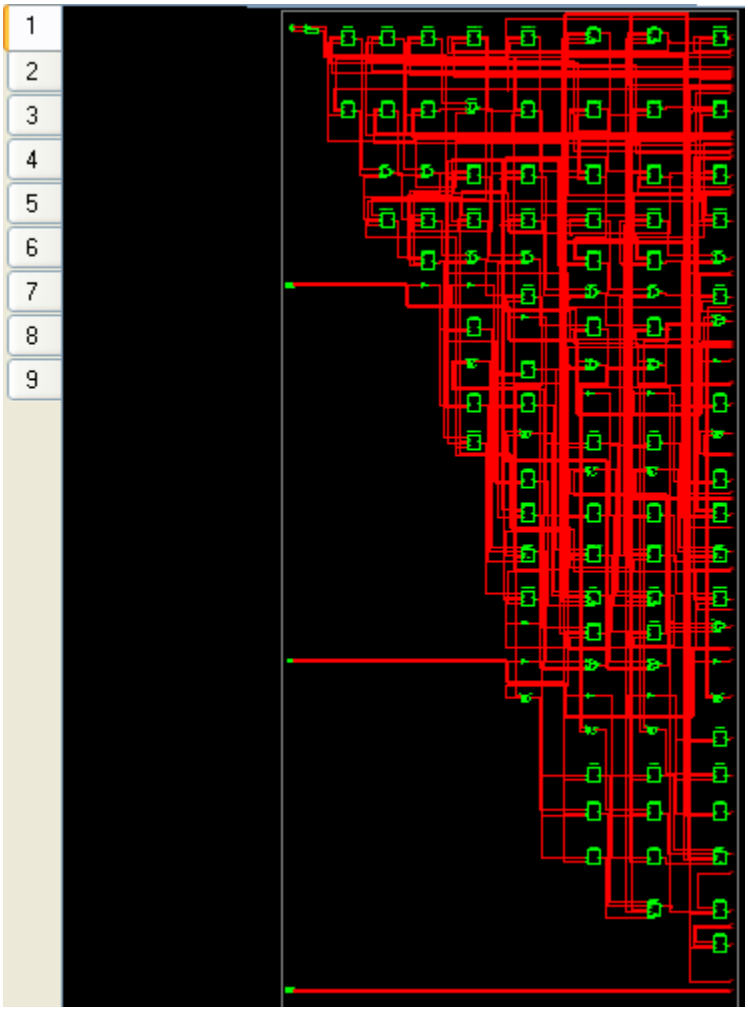
[13] R. Q. Zhang, ”Low Power CORDIC IP Core Implementation Project Report 2004/2005, Institute for System Level Integration”.

[14] A.Özdemir, K. Danışman, “FPGA Üzerinde Hibrit Sayısal Sistem Modellemesi, Erciyes Üniversitesi Elektrik Elektronik Mühendisliği Bölümü”.

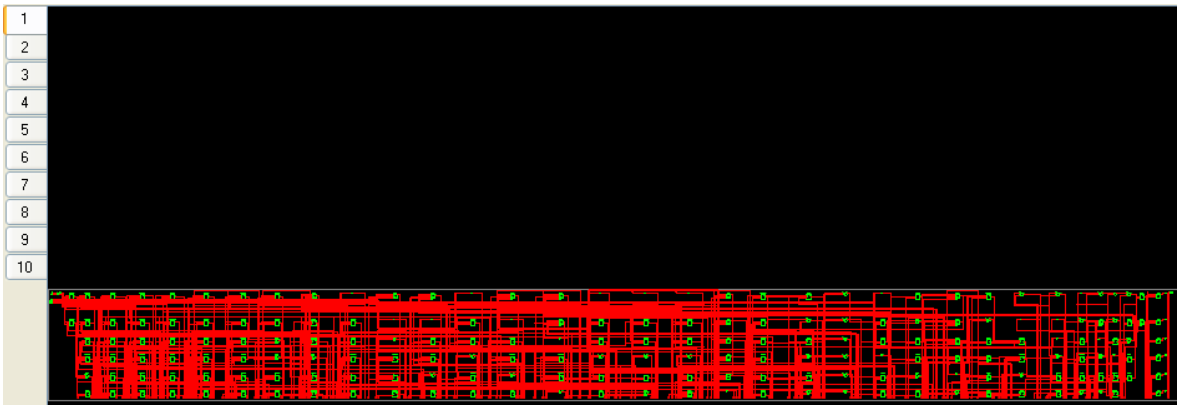
[15] Fabian Lis and George Pan, “Speeding up the CORDIC algorithm with a DSP,” Analog Devices, Inc., September 2008.

[16] Jason Todd Arbaugh, “Table Look-Up CORDIC: Effective Rotations Through Angle Partitioning,” The University of Texas at Austin, December 2004.

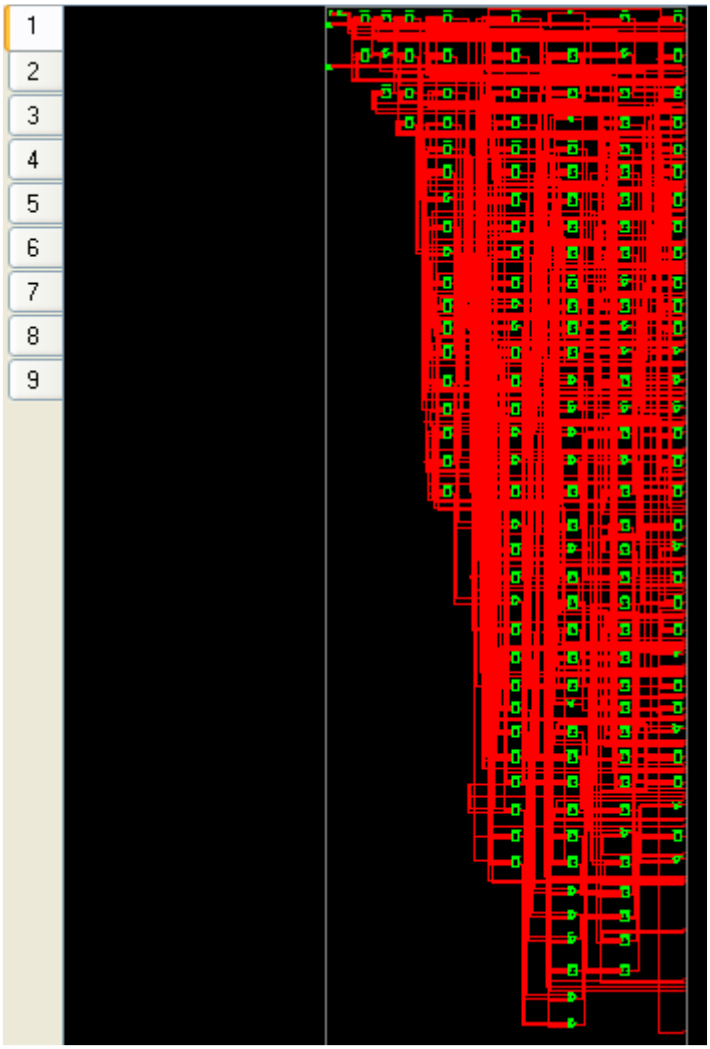
EK-A TEKNOLOJİ ŞEMATİK GÖRÜNTÜLERİ



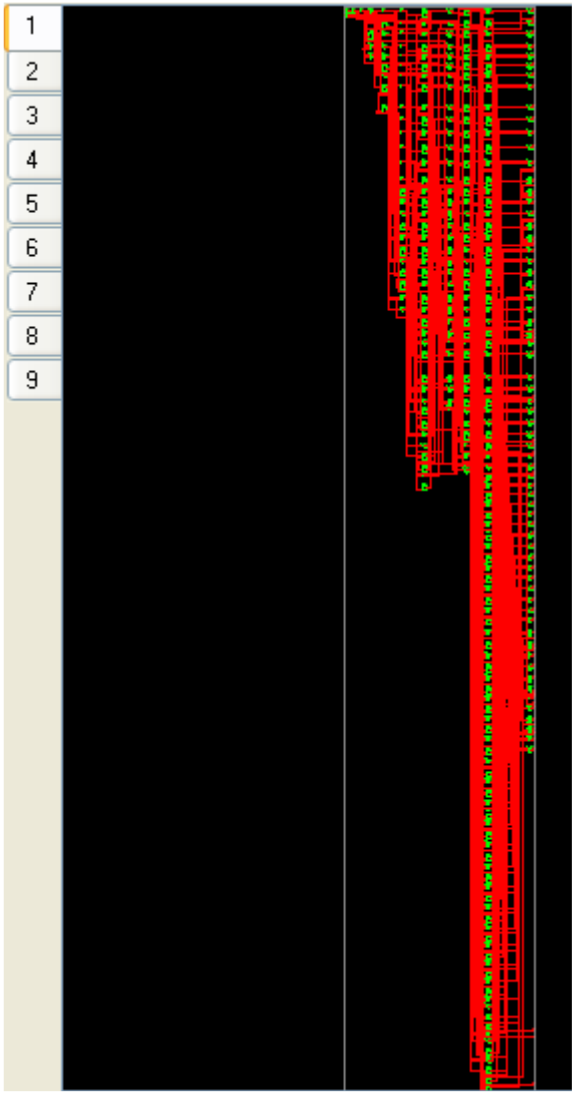
Şekil A- 1. Teknoloji şematik-sayfa1 (Klasik CORDIC)



Şekil A- 2. Teknoloji şematik-sayfa1 (Düşük Güç Tüketen CORDIC)



Şekil A- 3. Teknoloji şematik-sayfa1 (HARICP)



Şekil A- 4. Teknoloji şematik-sayfa1 (Pipeline Klasik CORDIC)

ÖZGEÇMİŞ

Kişisel Bilgiler:

Adı Soyadı : Emrah AYTÖRE

Doğum Yeri : Kadıköy / İSTANBUL

Doğum Yılı : 1982

Medeni Hali : BEKAR

Elektronik Posta : emrahaytore@gmail.com

Eğitim ve Akademik Durumu:

Lise : 1996-2000 Seyranbağları Anadolu Lisesi, ANKARA

Lisans : 2001-2005 Uludağ Üniversitesi, Elektronik Mühendisliği Bölümü, BURSA

Yabancı Dil:

İngilizce

İş Tecrübeleri:

2007....-.... ESDAŞ AŞ., ARGE Mühendisi