T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİR MOBİL ROBOTUN GERÇEK ZAMANLI MODELLENMESİ VE KONTROLÜ

YÜKSEK LİSANS TEZİ

Müh. Mustafa GEYİK

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Donanım

ŞUBAT-2010

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİR MOBİL ROBOTUN GERÇEK ZAMANLI MODELLENMESİ VE KONTROLÜ

YÜKSEK LİSANS TEZİ

Müh. Mustafa GEYİK

(03229101)

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Donanım

Tez Danışmanı: Yrd. Doç. Dr. Hayrettin CAN

Tezin Enstitüye Verildiği Tarih: 03 Şubat 2010

ŞUBAT-2010

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİR MOBİL ROBOTUN GERÇEK ZAMANLI MODELLENMESİ VE KONTROLÜ

YÜKSEK LİSANS TEZİ

Müh. Mustafa GEYİK

(03229101)

Tezin Enstitüye Verildiği Tarih : 03.02.2010
Tezin Savunulduğu Tarih : 19.02.2010

Tez Danışmanı : Yrd. Doç. Dr. Hayrettin CAN (F.Ü)
Diğer Jüri Üyeleri : Prof. Dr. Hasan Alli (F.Ü)
Yrd. Doç. Dr. Mehmet KARAKÖSE (F.Ü)

ŞUBAT-2010

ÖNSÖZ

FPGA (Field Programmable Gate Arrays; ”Programlanabilir Kapı Dizileri Alanı”)’lar sayısal tasarımlarda kullanılan farklı mantık kapıların milyonlarcasının tek bir entegre üzerinde toplanmasıyla oluşan yapılardır. Bu mantık kapılar istenildiği gibi programlanabilir ve istenilen her türlü sayısal tasarım bu elemanlar üzerinde gerçeklenebilir. FPGA’lar; donanım tabanlı sistemlere göre daha esnek yazılım tabanlı sistemlere göre daha hızlı sayısal tasarım ortamı oluşturarak bu iki sistem arasındaki boşluğu dolduran “Tekrar Düzenlenebilir Sayısal Sistemler” arasında önemli yer almaktadır.

Bu tez çalışmasında yüksek hareket kabiliyetlerinden dolayı oldukça geniş kullanım alanları bulan omni directional mobil robotlar için robot modellemesi ve mobil robotun belirtilen yörüngede hareket etmesi için gerekli mobil robot kontrolünün FPGA kullanarak gerçek zamanlı modellemesi yapılmıştır.

Özellikle yüksek hızlı işlem gerektiren uygulamalarda FPGA’lar sistem performansını oldukça artırmaktadır. Mobil robot modellemesi ve kontrolü süresinde yüksek hızlı veri işlemek için FPGA’a kullanarak yüksek verim alınması amaçlanmıştır.

Bu tez çalışmasını yönlendiren ve desteklerini esirgemeyen danışman hocam Sayın Yrd. Doç. Dr. Hayrettin CAN’a ve her türlü yardımlarından dolayı Erkan DUMAN’a teşekkür ederim.

Mustafa GEYİK
ELAZIĞ – 2010

İÇİNDEKİLER

Sayfa No

Sayfa No

ÖZET

Gerçek zamanlı işlemin önemi son zamanlarda fark edilmiş ve birçok uygulamada kullanılmıştır. Örneğin; gerçek zamanlı modelleme, kontrol ya da veri işleme sistemi gibi uygulamalarda, yüksek seviyeli işlem verimliliğinden dolayı oldukça kullanışlıdır. Geleneksel olarak sayısal işaret işleme (DSP) algoritmaları, bazı düşük hızlı uygulamalar için genel amaçlı (programlanabilir) DSP yongalarını kullanarak işlevlerini yürütürler. FPGA geliştirme maliyetinin yüksekliği ve üretimden sonra tasarım değişikliği yapamama gibi dezavantajlardan kaçınarak özel işlevselliğin avantajlarını devam ettirir. Ayrıca FPGA, hem kart alanını hem de sistem gücünü muhafaza ederken en uygun yöntemi kullanarak şartlara uyumluluk ve tasarım esnekliği sağlar.

Omni directional mobil robot bir tür holonomic robottur. Araba gibi çok yaygın bir mobil robot (nonholonomical) ile karşılaştırıldığında, omni directional mobil robot; tekerlekleri biri birinden bağımsız aynı zamanda eş zamanlı hareket edebilme yeteneğine sahiptir. Omni directional mobil robotun manevra kabiliyeti ona literatürde çok geniş bir çalışma alanı sağlar.

Bu tez çalışmasında, öncelikle bir omni directional mobil robot ve kontrol sistemi Matlab'da tasarlanmıştır. Daha sonra, gerçek zamanda mobil robotun performansını test etmek için robot modeli ve kontrol sistemi FPGA geliştirme kartında tasarlanmıştır. Bu çalışmada, Altera Stratix-II GX FPGA geliştirme kartı kullanılmıştır.

Omni directional mobil robotu modelleme işlemi sürecinde, ters kinematik ve dinamik robot modeli geliştirildi. Ek olarak, omni-robot içerisinde kullanılan motorlar modellendi ve sistemin kapalı döngü kontrolünü yapan her motor için PI denetleyiciler tasarlandı.

Bu çalışmanın amacı, bir omni directional mobil robot modeli ve gerçek zamanda çalışan bir kontrol sistemi geliştirmektir. Kontrol sistemini içeren modelin çalışma zamanı, gerçek zamanda çalışması açısından oldukça uygundur. Bu çalışmada; omni directional mobil robot modeli ve kontrol sistemi, iki farklı yörünge kullanılarak gerçeklenen Matlab simülasyonuyla tatmin edici sonuçlar alınarak doğrulandı.

Anahtar Kelimeler: FPGA, Omni Directional Mobil Robot, Kontrol, Gerçek Zaman

SUMMARY

An Mobile Robot Modelling and Controlling in Real Time

The importance of real-time processing has been recently realized and recognized in many applications. In some applications, e.g. real time modeling, control or data processing system, it is very useful to have high levels of processing throughput. Traditionally, digital signal processing (DSP) algorithms are implemented using general-purpose (programmable) DSP chips for low-rate applications. Advancements in Field Programmable Gate Arrays (FPGAs) provide new options for DSP design engineers. The FPGA maintains the advantages of custom functionality while avoiding the high development costs and the inability to make design modifications after production. The FPGA also adds design flexibility and adaptability with optimal device utilization while conserving both board space and system power.

Omni-directional mobile robot is a kind of holonomic robot. Compared with more common car like (nonholonomical) mobile robot, omni-directional mobile robot has the ability to move simultaneously and independently in translation and rotation. The maneuverability of the omni-directional mobile robot makes it widely studied in literature.

In this thesis, we first modeled an omni-directional mobile robot and a control system in Matlab. Then, in order to test performance of the mobile robot in real time, the model of the robot and control system has been developed in FPGA development board. In this study, Altera Stratix-II GX FPGA development board is used.

During the modeling process of the omni-directional mobile robot, inverse kinematic and dynamic model of the robot has been developed. Additionally, the motors used in the omni-robot have been modeled. And also, PI controllers have been constructed for each motors in order to closed loop control of the system.

The aim of the project is to develop an omni-robot model and control system works in real time. The execution cycle of model including control system is fairly suitable for working in real time. In this study, the omni-robot model and control system has been verified with Matlab simulation for two different trajectories and the satisfactory results have been obtained.

Key Words: FPGA, Omni Directional Mobile Robot, Control, Real Time

ŞEKİLLER LİSTESİ

Sayfa No

Sayfa No

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

SEMBOLLER LİSTESİ

$^{m}\mathbf{F}$: Hareketli eksendeki robotun toplam momenti

$^{t}\mathbf{F}$: Tekerleklerin itme kuvvetini

$^{w}\mathbf{F}$: Sabit eksen sistemindeki kuvvetleri

$^{w}\dot{\mathbf{X}}$: Robotun sabit eksene göre hızı

$^{w}\ddot{\mathbf{X}}$: Robotun x ve y yönündeki ivmesi

$^{m}\dot{\mathbf{X}}$: Robotun hareketli eksene göre hızı

$\boldsymbol{\omega}$: Robotun açısal hızı

$\mathbf{V_x}$: Robotun x eksenine göre hızı

$\mathbf{V_y}$: Robotun y eksenine göre hızı

$\dot{\mathbf{q}}$: Tekerleklerin açısal hızı

$\ddot{\mathbf{q}}$: Tekerleklerin açısal ivmesi

$^{w}_{m}\mathbf{R}$: Dönme matrisi

$\mathbf{B}$: Katsayı matrisi

$\mathbf{M}$: Robotun ağırlık matrisi

$\mathbf{m}$:Robotun kütlesi

$\mathbf{L}$: Robotun ağırlık merkezi ve tekerlekler arasındaki mesafe

$\mathbf{r}$: Tekerlek yarıçapı

$\boldsymbol{\phi}$: Sabit eksen ile hareketli eksen arasındaki açı

$\mathbf{J}$: Robotun kütlesel atalet momenti

$\mathbf{n}$: Motor redüktörü dişli oranı

$\mathbf{E}$: Motor giriş gerilimi

$\mathbf{k_E}$: Motor zıt EMK sabiti

$\mathbf{k_M}$: Motor moment sabiti

$\mathbf{R}$: Motorun armatürü

$\mathbf{k_{lr}}$: $1/R$ sabit değeri

$\boldsymbol{\tau_L}$: Motor momenti

$\boldsymbol{\tau_M}$: Yük momenti

$\boldsymbol{c}$: Sönümleme katsayısı

$\dot{\phi}$: Motorun dönme hızı

KISALTMALAR

ASIC : Application Specific Integrated Circuit
FPGA : Field Programmable Gate Array
DSP : Digital Signal Processing
PLD : Programmable Logic Device
PLA : Programmable Logic Array
PAL : Programmable Array Logic
CPLD : Complex Programmable Logic Device
SPLD : Simple Programmable Logic Device
MPGA : Mask Programable Gate Array
LUT : Lookup Table
LC : Lojik Cell
LE : Lojik Element
ALM : Adaptive Lojik Module
RAM : Random Access Memory
SRAM : Static Random Access Memory
ROM : Read Only Memory
CMOS : Comlimentary Metal Oxide Semiconductor
CAD : Computer Aided Design
G/Ç : Giriş/Çıkış
HDL : Hardware Description Language
VHDL : Very High Speed Application Specific Integrated Circuit Hardware Description Language
JTAG : Joint Test action Group
USB : Universal Serial Bus
PI : Proportional Integral
FIR :Finite Impulse Response
FFT :Fast Fourier Transform
VCO : Voltage Controlled Oscillator
IEEE : Institute of Electrical and Electronics Engineers

1.GİRİŞ

Sayısal sistemler; sayısal verileri işlemek için tasarlanmış yapılardır. Bu yapılar, sayısal verileri işlemek için hızlı donanımlara ve bu donanımlara işlev kazandıracak esnek yazılımlara ihtiyaç duyarlar. Sayısal sistemleri oluşturmada, donanım ve yazılım tabanlı olmak üzere iki geleneksel yöntem mevcuttur.

Donanım tabanlı yöntemlerde, sayısal verileri işlemek için ağırlıklı olarak ASIC (Application Specific Integrated Circuit; Uygulamaya Özel Tümleşik Devre) ve üniversal tümleşik devreler kullanılır. Her iki devre türü de özel bir fonksiyonu gerçekleştirmek için oluşturulduğundan, bu fonksiyonları etkin ve hızlı bir şekilde gerçekleştirirler. Ancak bu devrelerin gerçekleştirdiği fonksiyonları değiştirmek ve yenilerini eklemek için, bu devrelerin ve kullandıkları kartların tekrar tasarım ve üretim aşamalarından geçmesi gerekmektedir. Bu oldukça pahalı ve uzun bir süreç demektir.

Yazılım tabanlı yöntemlerde ise; tasarım değişikliklerinde daha esnek olan mikroişlemciler kullanılır. Mikroişlemcilerin çalıştırdığı yazılımlar değiştirilerek, hiçbir donanım değişikliğine gidilmeden tasarım ortamına yeni fonksiyonlar eklenebilir. Bu yöntemde, fonksiyonun gerçeklenmesi için komutların bellekten okunması, onları yorumlanıp uygulanması, sistemin hız ve performansını oldukça düşürür [1].

Günümüzde, sayısal sistemlerin kullanıldığı alanlardaki hızlı değişimler, üretimleri tamamlandıktan sonra da esnek ve genel amaçlı olacak şekilde tasarlanan sayısal sistemleri ortaya çıkarmıştır. Tekrar Düzenlenebilir Sayısal Sistemler (Reconfigurable Computing Systems) olarak adlandırılan bu sistemler, esnek ve genel amaçlı yapıları sayesinde yeni bir üretim aşamasına ihtiyaç duymadan, değişen sistem özelliklerine ve kullanıcı ihtiyaçlarına kısa sürede cevap verebilir. Bu özellikleri sayesinde bu sistemler, donanım tabanlı sistemlere göre daha esnek, yazılım tabanlı sistemlere göre daha hızlı sayısal tasarım ortamı oluşturarak, bu iki sistem arasındaki boşluğu doldururlar. Tekrar düzenlenebilir sayısal sistemlerin; ihtiyaç duyduğu esnek donanımlar FPGA (Field Programmable Gate Arrays; ”Programlanabilir Kapı Dizileri Alanı”) kullanılarak karşılanır [9,10]. FPGA, programlanabilir mantık blokları, bu blok dizisini çevreleyen giriş-çıkış blokları ve ara bağlantılar olmak üzere düzenlenebilir üç ana bölümden oluşur. Programlanabilir mantık blokları, ara bağlantılar içerisine gömülü şekilde bulunur. Programlanabilir mantık bloklarının yapılandırılması ve bu bloklar arasındaki iletişim ara

bağlantılar sayesinde gerçekleşir. Giriş çıkış blokları, ara bağlantılar ile bütünleşmiş devrenin paket bacakları arasındaki ilişkiyi sağlar [3, 12].

Kullanıcının tasarladığı devreye göre, lojik bloklar ve aralarındaki bağlantılar programlanır. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı örnek üretme özelliği FPGA'ları sayısal tasarım ortamlarının vazgeçilmezi haline getirmiştir.

FPGA'ların kullanımını ve programlanmasını kolaylaştırmak üzere FPGA üretici firmaları tarafından arayüz yazılımları oluşturulmuştur. Bu arayüz programıyla bilgisayar ortamında oluşturulan tasarım FPGA'da çalışır hale getirilir [3]. Örneğin Altera Firması üretmiş olduğu FPGA'lar için Quartus II yazılım arayüzünü oluşturmuştur.

FPGA'lar günümüzde milyonlarca lojik kapı içerecek seviyeye ulaşmıştır. FPGA'daki lojik kapıların yanı sıra FPGA içerisine ayrıca hafıza ve çarpma bloklarının eklenmesi ile FPGA'ların uygulamalarda çok daha esnek ve verimli kullanımına olanak sağlamıştır. Günümüzde oldukça geniş kullanım alanları bulan FPGA'lar özellikle şifreleme, filtreleme gibi hızlı işlem gerektiren uygulamalarda, paralel işlem gerektiren özel işlemler ve gerçek zaman uygulamaları için kullanılır [2].

Bu tez çalışmasında gerçek zaman uygulaması olarak bir mobil robot modellemesi ve mobil robotun belirtilen yörüngede hareket etmesi için FPGA tasarımı yapılmıştır. Özellikle kontrol süresinde gerekli olan yüksek hızlı veri işlemek için FPGA'ya başvurularak yüksek verim alınmaya çalışılmıştır.

Mobil robotların endüstriyel, medikal ve teknik uygulamalardaki önemi sürekli artmaktadır. Genelde robotlar insanların ihtiyaçlarını karşılayan ve insanların yaptıkları işleri daha verimli şekilde yerine getiren akıllı makineler olarak tanımlanabilir [4,21]. Mobil robotlar taşıma, kontrol, gözetleme gibi oldukça geniş kullanım alanlarına sahiptirler. Mobil robotlar omni directional hareketle çok yönlü hareket edebilme kabiliyetine sahiptirler ve bu yönleriyle geleneksel robotlara büyük üstünlük sağlarlar [8].

Omni directional mobil robotun tekerlekleri birbirinden bağımsız hareket ederek robota esnek hareket edebilme olanağı sağlamaktadır [5,7]. Omni directional mobil robota bu esnek hareketi özel teker yapıları sağlar. Bu tekerleklerin yüzeyi serbest dönen küçük silindirlerle (rollers) kaplıdır. Tekerlek merkezinin bir motor tarafından hareket ettirilirken, tekerlek yüzeyindeki küçük silindirlerin bu motor tarafından kontrol edilmemesi önemli bir yapı oluşturur[6,14].

Mobil robotun hareketi kinematik model ve dinamik model olmak üzere temel iki model üzerine kuruludur. Kinematik model ile robot tekerlekleri arasında hız bağıntısı kurularak robotun hızından tekerleklerin açısal hızları elde edilir, dinamik model ile kinematik modelde hesaplanan tekerlek hızları için gerekli motor moment (tork) bağıntıları ve bu moment değerini yakalayabilmek için motora uygulanması gereken gerilim bağıntıları elde edilir [6,7,15].

Omni directinal mobile robotlar üzerine 60'lı yıllarda yoğun şekilde çalışılmaya başlanmıştır. Günümüzde bu mobil robotlar üzerine birçok çalışma yapılmaktadır. Bu çalışmalar ilk başta omni directional mekanizmasının gerçeklenmesi üzerine yoğunlaşmıştır. Daha sonra dinamik model ile birlikte çalışmalar devam etmiştir. Watanabe üç tekerlekli mobile robot (tekerlekler ağırlık merkezine simetrik) ile dinamik model üzerine değişik çalışmalar yaparak üç tekerlekli mobile robotun dinamik denklemlerini geliştirmeye çalışmıştır [22]. Yine Carter, tekerleklerin ağırlık merkezine göre simetrik olmayan robotların modelleri üzerine çalışmıştır [24]. Omni directional mobil robotlar için ilginç çalışma alanlarından biri de "robocup" karşılaşmalarıdır. İlk kez 2000 yılında Avustralya Sidney'de yapılan robocup yarışları ile daha etkin hareket kabiliyetine sahip robotlar için omni directional robotları bu alanda kullanmak üzere çalışmalar yapıldı [16,23]. Yüksek hareket kabiliyetine sahip mobile robotların bu etkinliğini daha da artırmak ve hata oranını minumuma indirmek için omni directional mobile robotların farklı denetleyiciler ile hareketlerini düzenlemek için değişik çalışmalar yapılmıştır [19, 20, 22]. Bu tez çalışmasında 3 tekerlekli omni directional mobil robot modeli referans alınmıştır. Her bir tekerlek robotun ağırlık merkezine eşit mesafede olup aralarında 120^{o} açı farkı bulunmaktadır. Tekerlekler robotun altında bulunan üç motora bağlanmıştır.

Bu tez çalışmasında üç tekerlekli omni directional mobil robot modeli FPGA (Field Programmable Gate Arrays;" Programlanabilir Kapı Dizileri Alanı") ortamında kapalı döngü sistemi ile gerçeklenmiştir. Mobil robot hareketini daha da etkinleştirmek için her bir tekerlek için PI (Proportional Integral) denetleyici tasarlanmıştır. Model hesaplamalarında IEEE-754 32 bit floating point (kayan noktalı) sayılar kullanılmıştır.

İkinci bölümde; FPGA'ların yapısı, mimarisi, FPGA'larda tasarım sürecinin nasıl işlediği, FPGA tasarımı için Quartus arayüz yazılımı ve kullanımı, Quartusta PI denetleyici yapısı ve modelleme hesaplamaları için kullanılan kayan noktalı sayı sistemi hakkında bilgi verilmiştir.

Üçüncü bölümde; omni directional mobil robot yapıları, omni directional mobil robotların üstünlüğü, mobil robotun kinematik ve dinamik modelleri anlatılmıştır. Kinematik modelde; robotun hızından robotun tekerlek hızlarını hesaplamak için gerekli denklem ve bağıntılar verilmiştir. Dinamik modelde; kinematik modeldeki denklemlerden elde edilen tekerlek hızları için gerekli moment (tork), gerilim hesaplamaları için denklem ve bağıntılar verilmiştir.

Dördüncü bölümde; gerçekleştirilen FPGA tasarımı bloklara ayrılarak her bir bloğun yapısı ve işlevi detaylı olarak açıklanmıştır.

Beşinci bölümde; FPGA ortamında gerçeklenen gerçek zamanlı modelden alınan sonuçlar, Matlab ortamında yapılan simülasyonlar ile karşılaştırılarak gerçeklen tasarımın doğruluğu gösterilmiştir.

2. PROGRAMLANABİLİR KAPI DİZİLERİ ALANI

Sayısal tümdevre sürecinde 80'li yıllarda belli boşluklar görülmeye başlandı. Bir tarafta SPLD ve CPLD gibi programlanır yongalar vardı. Bunlar yüksek yapılandırılabilme, hızlı tasarım ve değişiklik sürelerine sahiptiler ama geniş ve karmaşık tasarımları destekleyemiyorlardı. Diğer tarafta ASIC tasarım bulunmaktaydı. Çok geniş ve karmaşık işlevleri desteklemelerine rağmen, oldukça pahalı ve zaman harcayan sürece sahiplerdi. Üstelik tasarımın geri dönüşü yoktu. Bu aralığı doldurmak amacıyla Xilinx firması FPGA adını verdiği yeni bir tümleşik devre sınıfı geliştirdi ve 1984 yılında pazara sunulacak hale getirdi. İlk FPGA'lar CMOS tabanlı ve yapılandırma için SRAM hücreleri kullanıyordu. İlk örnekleri çok daha basit olmalarına rağmen temelde var olan mimari çoğu açıdan hala kullanılmaktadır.

FPGA'lar 1990'ların ilk yıllarında artan kapasiteleri sayesinde geniş veri işlemleri gerektiren haberleşme ve ağ ortamlarında kullanımı arttı. 90'ların sonlarına doğru tüketiciye yönelik, otomotiv ve endüstriyel uygulamalardaki kullanımları devasa büyüme sergiledi. FPGA'lar genellikle ASIC (Application Specific Integrated Circuit; Uygulamaya Özel Tümleşik Devre) tasarımların ilk örnekleri veya yeni bir algoritmanın fiziksel gerçeklenebilirliğini doğrulamak adına donanım ortamı sağlamak için kullanılırlar. Bununla birlikte, düşük geliştirme maliyetleri ve kısa sürede pazara sunulabilme özellikleriyle gittikçe son ürün yelpazesindeki yerlerini almaktadırlar.

2000'lerin ilk yıllarında milyonlarca kapı içeren yüksek performanslı FPGA'lar piyasada yerini aldı. Bazıları gömülü mikroişlemci çekirdekleri, yüksek hızlı Giriş/Çıkış arayüzleri, gömülü RAM ve DSP (Digital Signal Processing) öbekleri sunmaktadır [1,10].

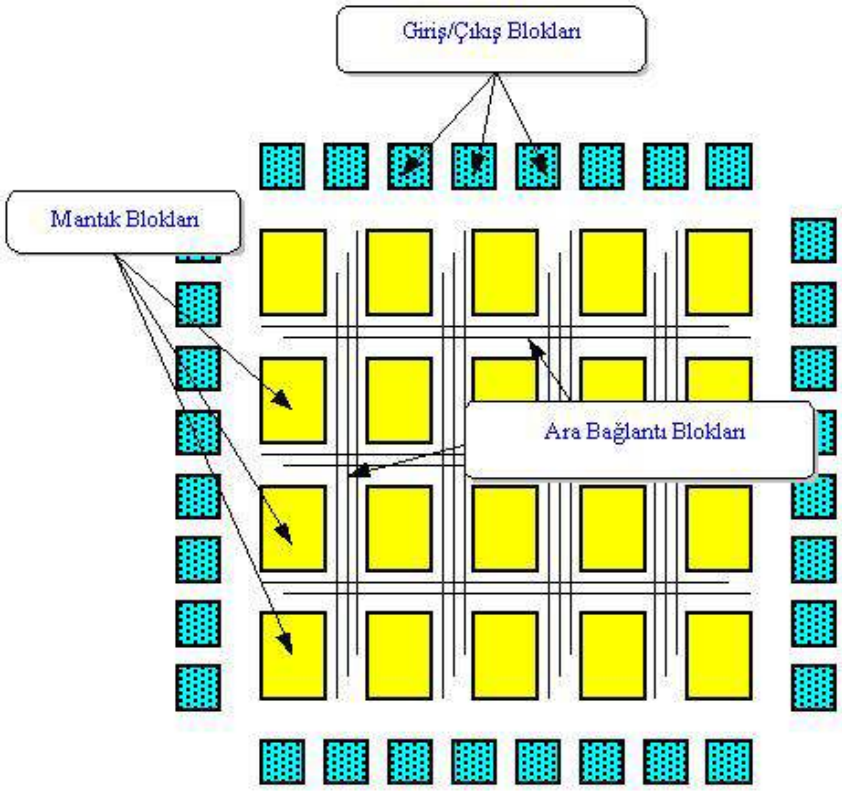
2.1. FPGA Mimarisi

2.1.1. FPGA Temel Bileşenleri

Genel olarak FPGA' lar üç ana birimden oluşur

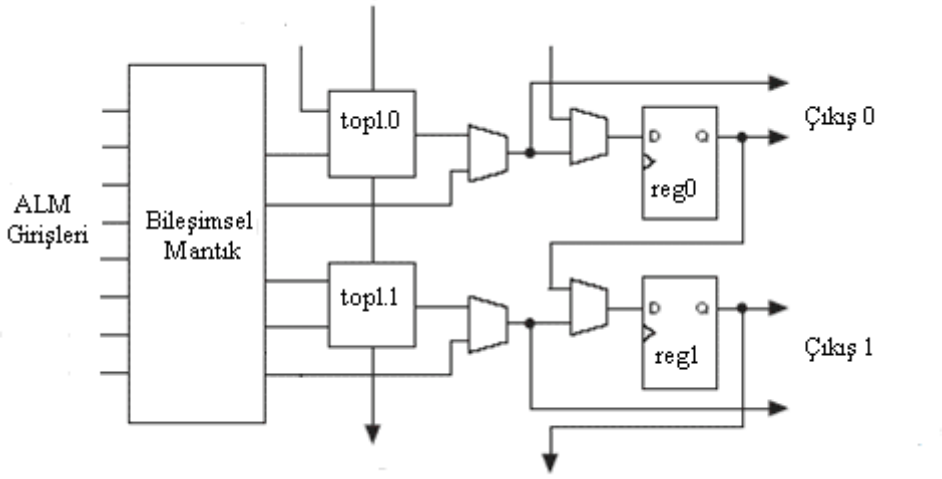
- Mantık Blokları
- Ara Bağlantı Blokları
- G/Ç Blokları

Şekil 2.1'de FPGA blokları gösterilmiştir.



Şekil 2.1. Genel FPGA yapısının üstten görüntüsü

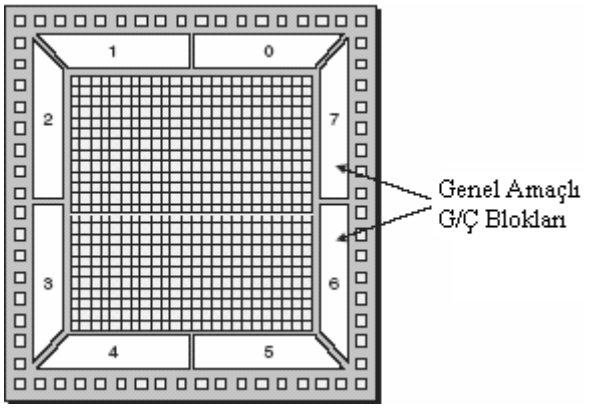
Mantık blokları; boolen fonksiyonlarının gerçeklendiği yapılardır. Her üretici farklı isimler kullanır. Örneğin Xilinx FPGA örneklerinde Mantık blokları "Lojik Cell (LC)" olarak isimlendirirken, Altera FPGA'ların Mantık bloklarını "Lojik Element (LE)" olarak isimlendirir, Stratix ailesiyle bu isimlendirme "Adaptive Lojik Module (ALM)" olmuştur. Şekil 2.2'de Stratix II ALM mantık bloğunun yüzeysel görünümü gösterilmiştir.



Şekil 2.2. Stratix II ALM yüzeysel görünümü

FGPA'larda programlanır mantık blokları arasındaki iletişim ve bu programlanır blokların istediğimiz işlev doğrultusunda yapılandırılması ara bağlantılar sayesinde gerçeklenir. Bir FPGA aygıtı içerisinde ara bağlantıların kapladığı alan %80'leri bulmaktadır. Dolayısıyla üreticiler bu büyük kütlenin en verimli şekilde yerleşimi için büyük çaba sarf etmektedirler. Altera; ALM (veya LE), bellek blokları, G/Ç bacakları ve varsa DSP blokları arasındaki bağlantıyı "MutiTrack" bağlantı yapısı olarak isimlendirdiği, "DirectDrive™" teknolojisi ile yapar. Multitrack bağlantılar, bloklar içi ve arasında sürekli ve başarıma göre yapılandırılan farklı uzunluk ve hızdaki yönlendirme hatlarını içerir.

G/Ç Blokları; gereken standarda uyacak şekilde sinyal alma ve üretme için yapılandırılabilir. Bu genel amaçlı G/Ç sinyalleri belli kümelere bölünmüştür. Her küme, özel G/Ç standardını desteklemek için ayrı ayrı yapılandırılabilir [10]. Şekil 2.3'de FPGA'daki genel amaçlı G/Ç blokları gösterilmiştir



Şekil 2.3. Genel amaçlı G/Ç Blokları

2.1.2. FPGA'ların Programlama Teknolojileri

FPGA yongaları farklı teknolojileri baz alarak üretilebilir. Yonga üreticileri farklı tekniklerde uzmanlaşmıştırlar. Programlama teknolojilerine göre FPGA'lar SRAM, antifuse, EEPROM/FLASH temelli aygıtlar olmak üzere üç temel guruba ayrılır.

2.1.2.1. SRAM Temelli Aygıtlar

FPGA' larda çoğunlukla SRAM temelli yapılandırma hücreleri yaklaşımı kullanılır. Bu tekniğin ana üstünlüğü yeni tasarım fikirlerinin çabucak geliştirilebilir ve sınanabilir olmasıdır. Bu sayede, standartlar ve protokoller geliştirirken daha kolay uzlaştırılabilir. Ayrıca düzeneğe ilk güç verildiğinde başlangıçta öz sınama (Self Test), kart/sistem sınaması gibi işlevleri gerçeklemesi için ve sonra ana görevi için programlanabilir. Yani sistem üzerinde programlanabilir. SRAM temelli yaklaşımın bir diğer üstünlüğü, teknolojinin ön saflarında yer almasıdır. FPGA sağlayıcılarının talepleri neticesinde bellekler üzerinde uzmanlaşmış firmalar bu alanda araştırma ve geliştirme için büyük kaynak harcamaktadırlar. Üstelik SRAM hücreler, yongadaki diğer birimlerle tamamen aynı CMOS teknolojisi ile üretildiklerinden özel işlem basamağı gerektirmezler.

Olumsuz tarafı ise sistemin her açılışında aygıtın tekrar yapılandırılmak zorunda olmasıdır. Bu durum harici bellek gibi fazladan masraf ve alan teşkil eden birimler gerektirir. Bir diğer olumsuzluğu büyük yönlendirme gecikmesidir [11].

2.1.2.2. Antifuse Temelli Aygıtlar

SRAM temelli yaklaşımdan farklı olarak devre dışında özel programlayıcılar ile programlanır. Bu yaklaşımda veriler uçucu değildir. Sistem güçten kesildiği zamanda yapılandırma verilerini saklaması harici bellek gereksinimini ortadan kaldırır.

Bir diğer önemli üstünlüğü ara bağlantı yapısının doğal olarak ışınım etkisine görece bağışıklı olmasıdır. Bu durum askeri ve uzay uygulamalarında özel ilgi sebebidir. Çünkü SRAM temelli bileşenler ışınıma maruz kalacak olursa hatalara yol açabilir. Antifuse bir kez programlandıktan sonra bu yolla değiştirilemez. Ancak aygıttaki herhangi bir flip-flop ışınıma hassasiyetini sürdürür. Bu yüzden yoğun ışınımlı ortamlar için flip-floplar "triple redundancy design" yöntemi ile korunmalıdırlar. Bu yöntem her yazmacın

üç kopyasının olması ve hata durumunda çoğunluktaki değerin hatalıyı düzeltmesi şeklindedir.

Muhtemelen en önemli özelliği yapılandırma verisinin FPGA içerisine gömülmesidir. Her antifuse işlendiğinde, aygıt programlayıcı o öğenin tamamen programlandığının tespitine kadar sınamasını devam ettirir. Sonrasında sıradaki antifuse öğesine geçer. Üstelik aygıt programlayıcıları yapılandırmanın başarılı biçimde gerçeklendiğini özdevimli olarak doğrulayabilirler. Bu özellik milyonlarca programlanır öğe içeren yongalar için önemlidir.

Antifuse öğesi SRAM temelli öğeye göre kendi başına daha az yer kaplıyor ve enerji harcıyor olmasına rağmen her öğe için fazladan programlama devresi gerekir. Bu nedenle bu açılardan önemli fark yaratmaz. Yönlendirme gecikmesi daha küçüktür. Bu antifuse tekniğini SRAM' e göre daha hızlı kılar [2,12].

2.1.2.3. EEPROM/FLASH Temelli Aygıtlar

SRAM karşıtlarına benzer olarak yapılandırma hücreleri birbirlerine uzun ötelemeli yazmaç biçimli zincirler şeklinde bağlanmışlardır. Bu yongalar aygıt programlayıcıları ile devre dışında programlanabilir. Bazı çeşitleri devre içinde de programlanabilir (In System Programmable). Ancak programlama hızları SRAM temelli bileşenlere göre yaklaşık üç kat fazladır. SRAM hücrelerine göre daha küçük olduklarından ara bağlantı gecikmeleri daha azdır. Ancak standart CMOS teknolojisine ilaveten yaklaşık beş işlem basamağı gerektirdiğinden SRAM temelli aygıtların birkaç nesil gerisinde kalır.

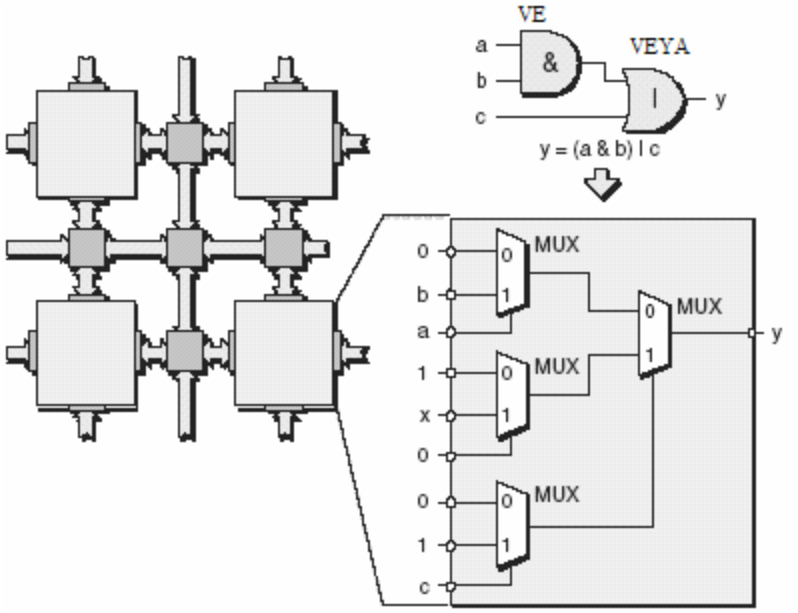
2.1.3. Hücre Mimarileri

FPGA'ları diğer aygıtlardan ayıran ana özelliği ağırlıklı olarak programlanır ara bağlantılar içerisine gömülü, çok sayıda kısmen küçük programlanır mantık öbekleri içermeleridir.

İki temel programlanır mantık öbeği düzenlemesi vardır: MUX tabanlı ve LUT tabanlı.

2.1.3.1. MUX Tabanlı

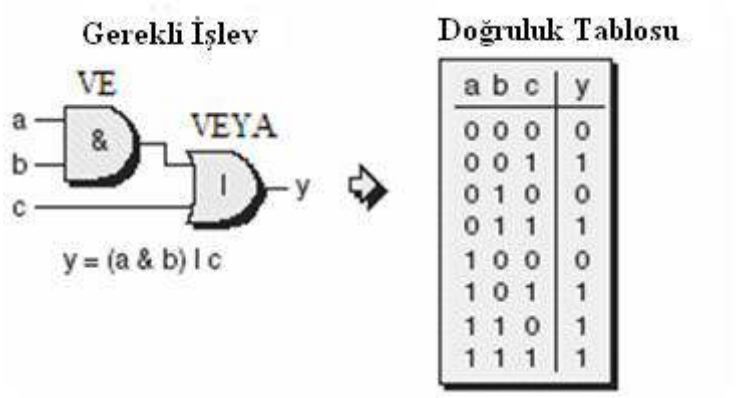
MUX tabanlı yapının temel bloğu MUX'ların çeşitli konfigrasyonlarından ve olabildiğince az VE, VEYA gibi lojik kapılardan oluşur. Ardışıl devrelerin gerçeklenebilmesi amacıyla MUX tabanlı yapı içerisinde veri tutucu flip-flop gibi devre elamanları kullanılabilmektedir. Aygıt mantıksal 0, mantıksal 1 veya diğer öbek ve asıl girişten gelebilecek sinyalleri asıl veya tümleyen değeri ile programlanabilir. Şekil 2.4'de MUX tabanlı mantık bloğu yapısı gösterilmiştir.



Şekil 2.4. MUX tabanlı mantık bloğu

2.1.3.2. LUT Tabanlı

Bir grup giriş sinyali başvuru çizelgesi (LookUp Table) için işaretçi olarak kullanılır. Çizelgenin içeriği, istenen değeri içeren her giriş birleşimini gösteren hücreler ile düzenlenir. Şekil 2.5'de üç girişli bir işlev ve doğruluk tablosu gösterilmiştir.



Şekil 2.5. Gerekli işlev ve doğruluk tablosu

Bu işlev 3-girişli LUT'u uygun değerler ile yükleyerek yapılabilir. Örnek için LUT'ların SRAM hücrelerden oluştuğunu düşünelim. Kullanılan ana teknik istenen SRAM hücreleri seçmek için basamaklı iletim geçitlerini, girişleri kullanarak belirlemektir. Birkaç kademe derinliğindeki mantık kapıları için LUT yaklaşımı kaynak kullanımı ve giriş-çıkış gecikmeleri açısından daha verimli olacaktır. Günümüzde FPGA mimarisi çoğunlukla LUT tabanlıdır [10].

2.2. FPGA ve DSP Kıyaslaması

DSP tipik olarak C yada assembly kodu ile programlanan özelleştirilmiş bir işlemcidir. Özellikle şartlı işlemli (conditional processing) kompleks matematik tabanlı işlemler için uygundur. Saat hızı (clock rate) ve saat başına icra edebileceği kullanışlı operasyonların sayısı ile çalışması sınırlıdır.

Buna karşın bir FPGA entegresi çarpan, kayıtçılar, toplayıcılar ve benzerlerini oluşturmak için kapıların birlikte oluşturulması ile programlanır. FPGA'ların performansı saat hızı (clock rate) ve sahip olduğu kapıların limitleri ile sınırlıdır. FPGA'lar DSP'nin yapmış olduğu işlemleri daha hızlı gerçekleştirebilecek çarpma blokları içerir.

Örnekleme oranı birkaç Mhz'in üzerinde olduğunda bir DSP veriyi kayıpsız transfer etmek için çok fazla çalışması gerekir. Bu işlemcinin hafıza yolları gibi paylaşılmış kaynakları kullanma zorunluluğundan dolayıdır. Diğer taraftan bir FPGA veri alabilmek için lojik alan tahsis edebilir. Böylece yüksek I/O oranı sağlayabilir. Bir DSP harici hafızanın kullanımı için optimize edilmiştir, böylece işlemde büyük bir veri kümesi kullanılabilir. FPGA'lar sınırlı miktarda dahili depoya sahiptirler böylece daha küçük veri

kümeleri üzerinde işlem yaparlar. Ancak harici hafızalı FPGA modülleri bu kısıtlamayı kaldırmak için kullanılabilir.

Bir DSP işlem birimlerini basitçe tekrar kullanımı için tasarlanmıştır. Mesala bir çarpan FIR'ı hesaplamak için kullanılırken aynı çarpan FFT'leri hesaplayan başka bir rutin iş tarafından da kullanılabilir. Fakat bu iş FPGA'da zordur, ama genelde FPGA daha fazla çarpan içerir.

Yeni FPGA'ların hafıza bant genişliği, bir mikroişlemci veya bir DSP işlemcisinin çalışmasını, saat oranları bazında beş kattan on kata kadar daha ileri götürmüştür. FPGA'nın en önemli avantajlarından biri de esnek mimarisidir. Ancak bu esneklik donanım maliyeti getirir. Fazla esneklik fazla kapı anlamına gelir, bu da fazla silisyum alanı, fazla yönlendirme kaynakları ve yüksek enerji tüketimi demektir.

Sayısal işaret işleme alanında, bütün karakteristik özelliklerine rağmen FPGA kullanımının yaygınlaşmasını engelleyen bazı faktörler mevcuttur. Bunların başında DSP tasarımcılarının C veya assembly dilleri ile program yazma bilgilerine sahip olmaları, diğer taraftan bunlardan farklı yapıdaki VHDL veya Verilog gibi donanım tanımlama dillerine birçok avantaj sağlamalarına rağmen uzak kalmalarıdır.

FPGA'lar programlama kabiliyetleri yönüyle DSP ve mikroişlemcilere göre sınırlıdırlar. Bununla birlikte birçok ara yüz tanımlama araçları FPGA tasarımını desteklemekte ve bu sayede kodlanan devreler, lojik elemanlarıyla bir devre çizimi olarak gösterilmektedir. VHDL' in tasarımcılara verdiği destekle tasarımlar çok daha çabuk ve etkili bir şekilde gerçekleştirilir.

Bazı yüksek performanslı işaret işleme uygulamalarında FPGA'nın paralel mimari avantajı DSP'den daha verimlidir. Aynı zamanda, yonga seviyesinde enerji tüketimleri yüksek olmasına rağmen FPGA' ların toplam enerji tüketimi DSP işlemcilere göre daha az olabilir. Ancak bu karşılaştırmaların çok kaba tahminlere dayalı olduğu unutulmamalıdır. Çoğu yüksek performanslı işaret işleme uygulamalarında FPGA'lar enerji etkinliği konusunda DSP'ye göre daha avantajlıdır.

Bir uygulamada FPGA veya DSP seçimi için şu kriterlere dikkat edilmelidir:

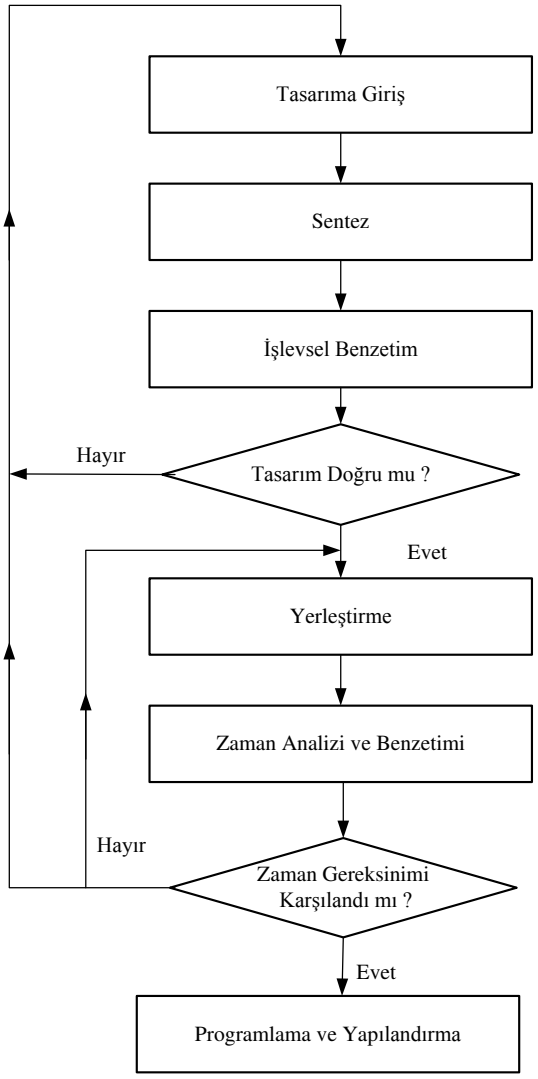
1. Sistemin parçasının örnekleme oranı birkaç Mhz'in üzerinde ise FPGA doğal bir seçim olacaktır.
2. Sistem C'de kodlanmışsa DSP direk olarak bu sistemi gerçekleştirebilir. Belki en iyi performans bu olmasa bile hızlı çalışır.

3. Sistemin veri hızı oranı saniyede 20–30 Mbyte/saniye ise FPGA daha iyi çözüm verir.
4. Çalışmaya uygun kütüphaneler hangisinde varsa o tercih edilir.

Genelde çoğu sistem birçok bloktan oluşur. Bu blokların bazıları FPGA'da bazıları ise DSP de gerçekleştirilir. Düşük örnekleme oranları ve karmaşıklığı az olanlarda DSP kullanılması daha uygundur; yüksek örnekleme oranları, tekrarlı görevlerde FPGA kullanımı daha uygundur. Sonuç olarak FPGA ve DSP sinyal işlemeye iki farklı yaklaşımı temsil eder. Çoğu yüksek örnekleme oranında FPGA daha iyidir. En ideal sistem, işi FPGA ve DSP arasında bölmektir [26].

2.3. FPGA Programlama ve Uygulama Geliştirme

FPGA yongasında istediğimiz programı gerçeklemek için aşağıda verilen akış şeması takip edilir. CAD (Computer Aided Design) olarak bilinen bilgisayar destekli tasarım yazılımları sayesinde günümüzde tasarımcılar üzerinden büyük yük kalkmıştır. Tasarım aşamasında büyük zaman alabilecek basamaklar ortadan kaldırılıp tasarımcının özel yeteneği ile fark yaratılmaktadır. Şekil 2.6'da FPGA'da tasarım sürecinin akış diyagramı gösterilmiştir.



Şekil 2.6. FPGA kullanılarak yapılan tasarım sürecinin akış diyagramı

Tasarıma Giriş: İstenen devre şematik veya öbek çizimler ile yapılır. Artan kapı kullanımı sonucunda Verilog veya VHDL (Very High Speed Application Specific Integrated Circuit Hardware Description Language) gibi HDL (Hardware Description Language) donanım tanımlama dillerinden herhangi birini kullanmak kaçınılmaz olmuştur. Bu diller, mantıksal gerçeklemeler için özelleşmiştirler. Bu sayede elektronik bileşenlerin davranışsal ve yapısal tanımlaması kolayca yapılarak esnek ve hızlı tasarım yapılabilir. Aksi takdirde milyonlarca kapıyı tanımlamak mümkün olmazdı.

Sentez: CAD sentez araçları ile devremiz için gerekli olan mantıksal öğeler ve bu öğeler arasındaki bağlantılar oluşturulur.

İşlevsel Benzetim: Sentezlenen devre bu aşamada işlevsel doğruluğu açısından sınanır. Hata durumunda tasarımda değişiklik yaparak sentez ve benzetim basamakları tekrarlanır.

Yerleştirme: İlgili yerleştirme araçları, bağlantı listesinde tanımlanmış mantıksal öğelerin FPGA yongası içerisindeki gerçek mantıksal öğelere yerleşimi yapar. Mantıksal öğeler arasındaki gerekli bağlantılar için yönlendirme hatları belirlenir.

Zaman Analizi ve Benzetimi: Yerleştirilmiş devrenin çeşitli yolları arasındaki yayılım gecikmeleri analiz edilir. Bu sayede devre beklenen başarım için sınanır ve sonucunda gerekli bağlantılar arasındaki zaman gecikmeleri gösterilir. Zaman benzetiminde ise yerleştirilmiş devre zamanlama ve doğruluk açılarından sınanır. İstenen zaman gereksinimleri sağlanamaz ise tasarım sürecinde en başa dönülür ve tasarımda belirlenen hata ile ilgili gerekli geliştirme yapılır. İşlemler benzetim sonuçları yeterli oluncaya kadar tekrarlanır.

Programlama ve Yapılandırma: Son aşamada belirli donanımlar üzerinden, gerçek FPGA yongası arzu edilen devreyi gerçeklemek için programlanır. Yapılandırma ile mantıksal öğeler istenen şekilde yapılandırılır ve içerikleri belirlenir [12].

2.3. Quartus II Arayüz Yazılımı

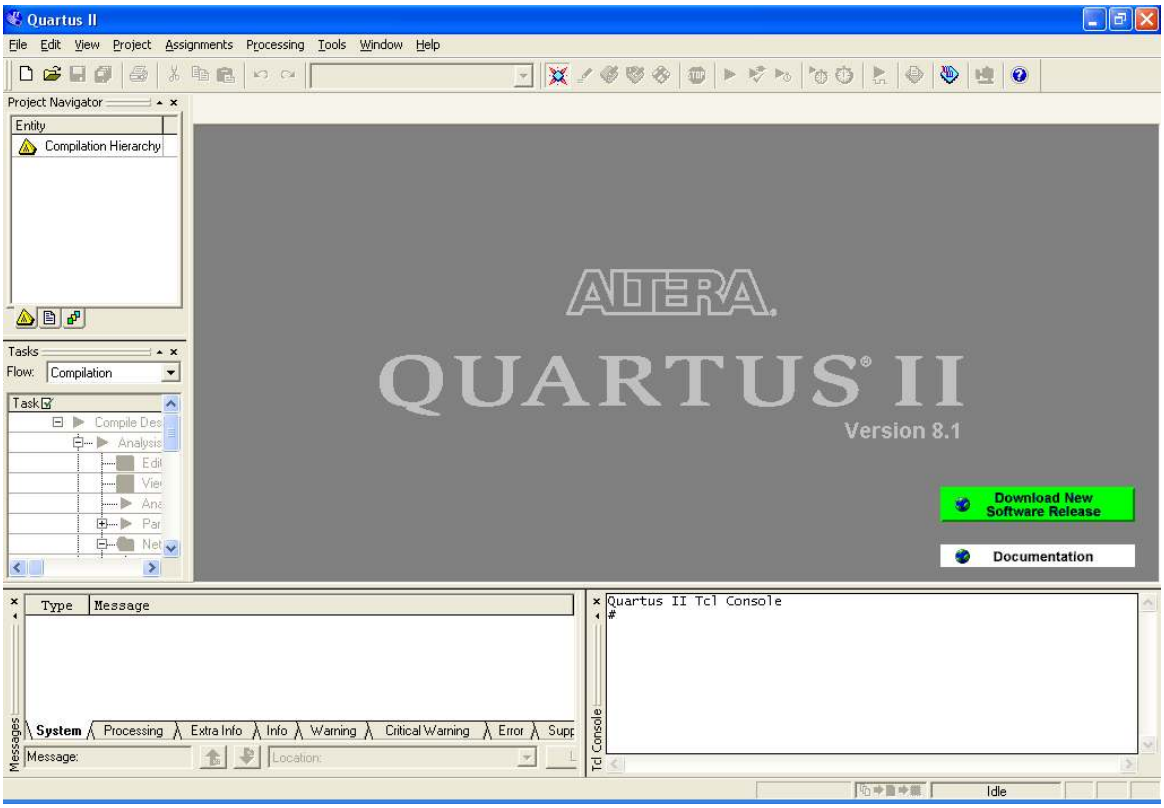
FPGA'ları programlamak için genellikle VHDL ve Verilog donanımsal tanımlama dilleri kullanılır. Fakat donanımsal tanımlama dilleri ile FPGA tasarımı gerçekleştirmek güç ve zaman alıcı olduğundan grafiksel arayüz blok tasarım yazılımları geliştirilmiştir. FPGA'ları programlamak için genel bir grafiksel arayüz blok tasarım yazılımı mevcut olmadığından her firma kendi FPGA'sı için kendi özel arayüz yazılım tasarımını gerçekleştirmektedir.

Günümüzde FPGA üretiminde iki büyük firma olan Altera ve Xilinx ön plana çıkmaktadır, FPGA piyasasının büyük çoğunluğunu bu iki firma elinde bulundurmaktadır. Bu firmalardan Altera firması kendi FPGA'ları için Quartus II arayüz yazılımını, Xilinx firması ise ISE arayüz yazılımını kullanmaktadır. Şekil 2.7'de Quartus II arayüz yazılımı gösterilmiştir.

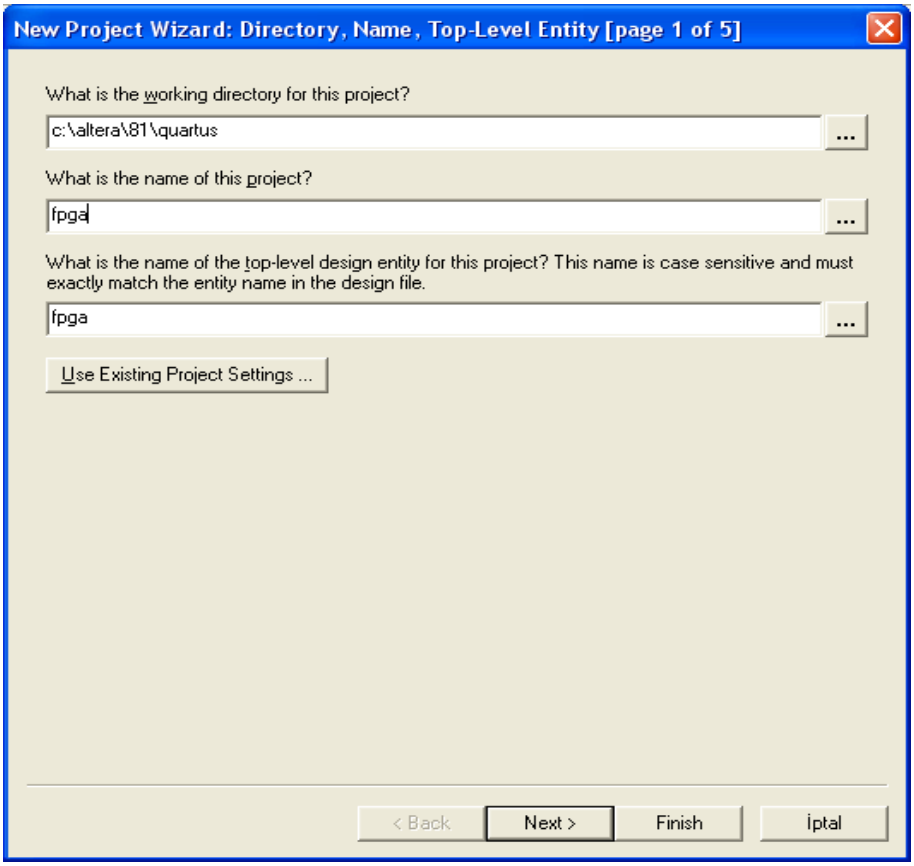
Bu tez çalışmasında Altera firmasının Stratix II FPGA'sı kullanıldığından oluşturulan FPGA tasarımı Altera firmasının Quartus II grafiksel arayüz blok tasarım

yazılımı ile gerçekleştirilmiştir. Aşağıda Quartus II'de bir proje tasarımının nasıl gerçekleştirildiği yüzeysel olarak açıklanmıştır.

Quartus II yazılımı kurulup lisans ayarlaması yapıldıktan sonra yeni bir proje oluşturmak için File menüsünden 'New Project Wizard' seçilir. Daha sonra Şekil 2.8'deki 'New Project Wizard' penceresi ekrana gelir. Ekrana gelen pencerede proje dosyalarının yer alacağı dizin, proje adı ve proje tasarım entity adı belirtilir.

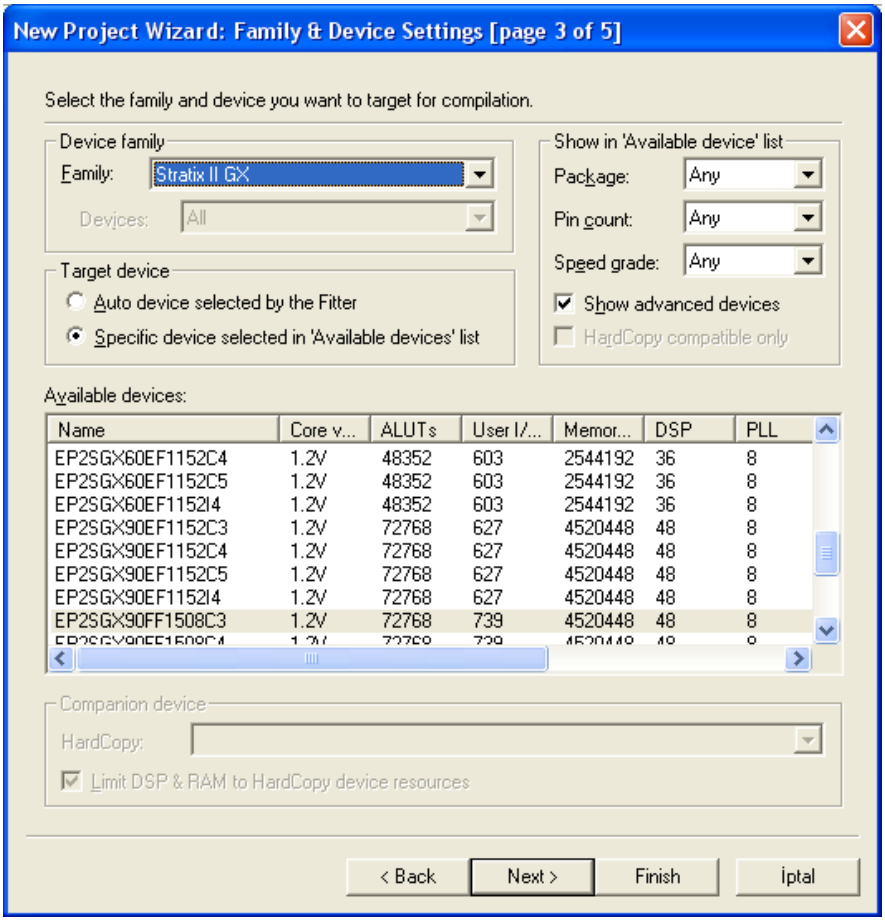


Şekil 2.7. Quartus II yazılım arayüzü



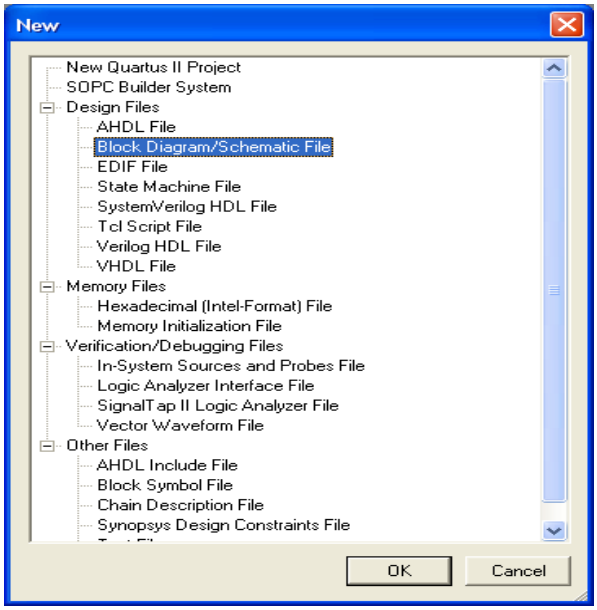
Şekil 2.8. 'New Project Wizard' ekranı

Daha sonraki aşamada proje için kullanılabilecek uygun FPGA seçme aşamasına geçilir. Şekil 2.9'da gösterildiği gibi FPGA aileleri listelenerek amaca uygun veya geliştirme kartı üzerinde bulunan FPGA seçilir. Sonra gelecek adımlar uygun cevaplarla geçilerek proje üzerinde çalışılmak üzere hazır duruma gelmiş olunur.



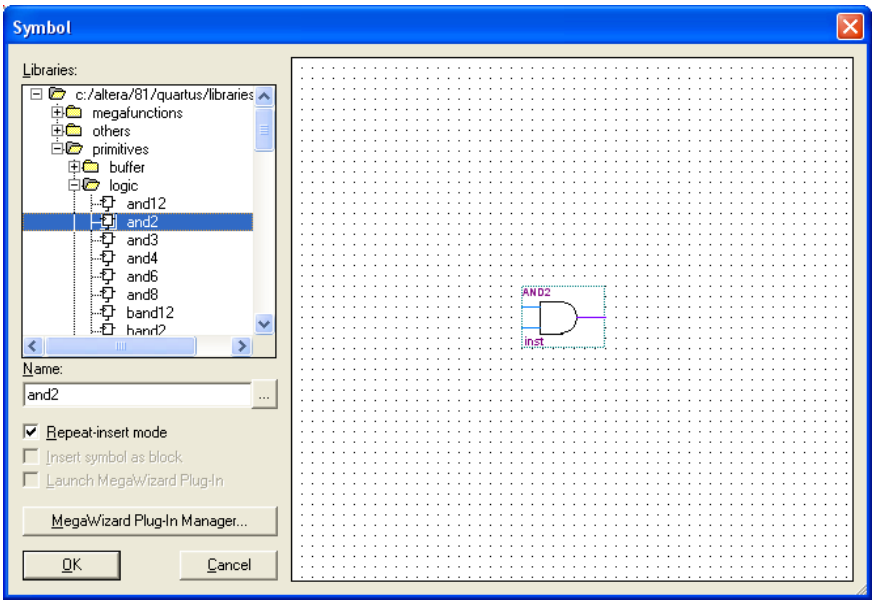
Şekil 2.9. 'New Project Wizard' FPGA seçme ekranı

Üzerinde çalışılmaya hazır hale gelen proje için öncelikle tasarım dosyası oluşturulması gerekir. Tasarım dosyası oluşturmak için file menüsünden 'New' seçildiğinde ekrana şekil 2.10'daki pencere gelir. Bu pencereden 'Block Diagram/Schematic' seçeneği seçilerek blok tasarımı için gerekli tasarım dosyası oluşturulur.



Şekil 2.10. Şematik tabanlı tasarım dosyası oluşturma

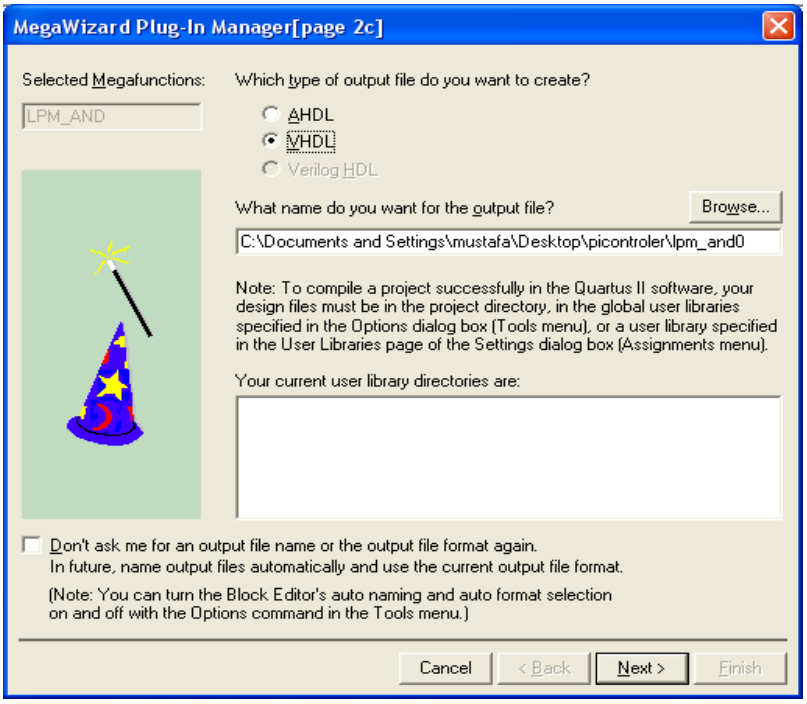
Şematik tabanlı tasarım dosyası oluşturulduktan sonra kullanıcı 'Symbol Tool' içerisinde bulunan macrofunctions, megafunctions, primitive kütüphane blok dosyaları kullanılarak tasarımı gerçekleştirmeye başlanır. Kütüphane dosyalarının kullanıcı için yetersiz gelmesi durumunda Altera tabanlı lpm ve alt fonksiyonları kütüphaneye eklenebilir. Şekil 2.11'de 2 girişli VE kapısının tasarım dosyasına eklenmesi gösterilmiştir.



Şekil 2.11. Tasarım dosyasına iki girişli VE kapısı eklenmesi

Tasarım adım adım oluşturulurken eklenen blokların arka planda hangi HDL dili ile tasarım dosyasının oluşturulacağı tasarımcı tercihine bırakılmıştır. Bu proje tasarımında

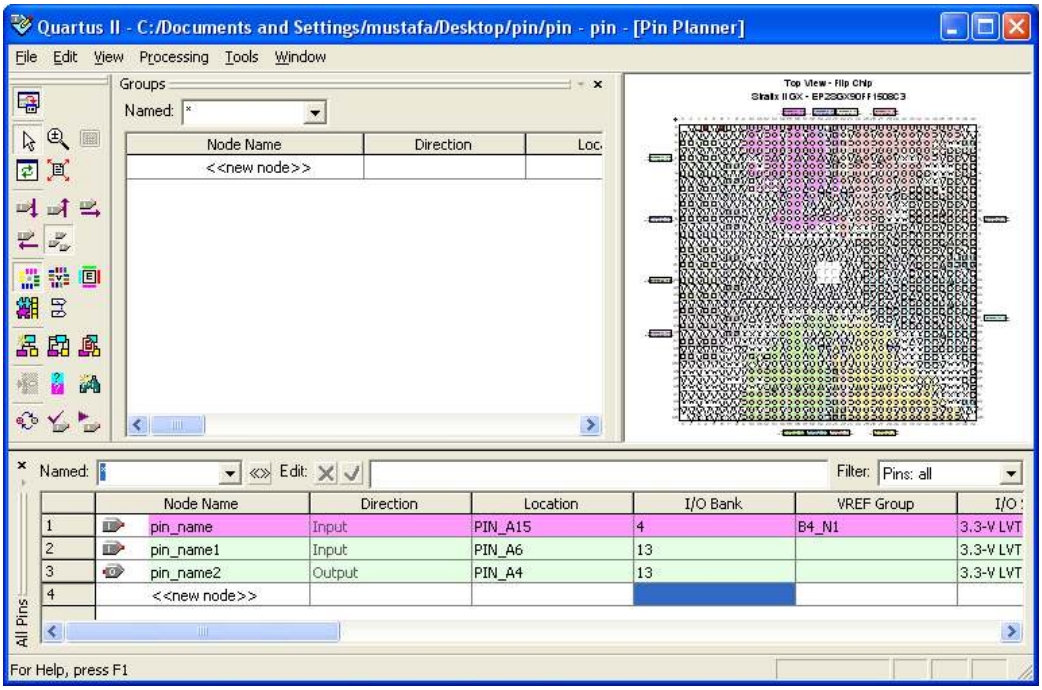
blok tasarım dosyaları VHDL dili ile oluşturulmuştur. Şekil 2.12'de iki girişli ve kapısının blok tasarım dosyasının oluşturulması VHDL dili tercih edilirken gösterilmiştir.



Şekil 2.12. VE kapısı blok dosyası için VHDL tercih edilirken

Proje tasarımı tamamlanıp derlendikten sonra oluşan .sof ve .pof uzantılı dosyalar USB Blaster veya JTAG yoluyla FPGA'ya gömülür.

Ayrıca tasarımın sonuçlarını tasarım dosyasını FPGA üzerine taşımadan önce Quartus II'nin kendi simülasyonundan sonuçlar değerlendirilebilir. Bunun için tasarıma 'Vektor Waveform File' simülasyon dosyasını ekleyip gerekli giriş ve çıkış pinlerini bu dosyaya kaydetmek gerekir. Tasarımda kullanılan pinlerin FPGA üzerindeki atamaları Assigment menüsünden "pin planer" ile yapılır. Şekil 2.13'de 2 girişli VE kapısına pin atamaları gösterilmiştir. Kullanıcının, tasarımı için kullanacağı uygun pinler FPGA kullanım katalogunda gösterilen etiketlere uygun olarak Quartus ortamında bağlantı atamaları yapılır.



Şekil 2.13. VE kapısına "Pin Planer" ile pin atama

2.4. Donanım Tanımlama Dilleri

Lojik devre tasarımının başlangıç dönemlerinde, tasarım lojik kapılar seviyesinde yapıldı. Ancak tasarımların karmaşıklaşması ve bunların lojik kapılar kullanılarak gerçeklenme zorluğu nedeniyle çeşitli yazılım dilleri tasarlanmaya başlandı. Bu dillere genel olarak HDL (Hardware Description Languages) adı verilir. Bu dillerden en çok kullanılanlardan biri VHDL yazılım dilidir. VHDL yazılım dili kullanılarak, her türlü lojik devre tasarlanabilir. Tasarımın işlevi yazılım programı içerisinde bulunan belirli test bölümleri vasıtasıyla kontrol edilebilir. Bu sayede sistem, donanım yapısı kurulmadan bilgisayar ortamında test edilebilir. Testlerin donanıma ihtiyaç duyulmadan yapılabilmesi tasarım süresinin kısalmasını sağlar.

VHDL'den başka en çok kullanılan donanımsal tanımlama dillerinden biride Verilog'dur. Verilog dili C programlama diline yakın bir söz dizimine sahiptir. Verilog geleneksel programlama dilleri gibi basamaklarını tam olarak ardışık bir şekilde yürütmez. Verilog tasarımı modüller arasında bir hiyerarşi bulundurur. Modüller bir takım giriş, çıkış ve çift yönlü portlar şeklinde tanımlanır.

2.4.1. VHDL dili mimari yapıları

VHDL dili 3 mimari yapıdan oluşur. Bunlar aşağıda açıklanmaktadır.

2.4.1.1. Davranışsal mimari

Davranışsal mimaride, sistemin yapması gereken işlemler, 'process' yapıları kullanılarak yapılır. Ancak tasarımın nasıl gerçekleneceği konusunda bilgi verilmez. VHDL programı derleyicisi, varsa uyarılarını ya da hatalarını, tasarımın ne kadar yer kaplayacağını kullanıcıya bildirir. Bu nedenle bazı durumlarda programda yapılan küçük değişimler, gerçekleme alanı bakımından büyük değişiklere neden olabilir. Bunun için tasarım yapılırken sürekli olarak donanım düşünülmelidir.

Davranışsal mimaride 'process' ile birlikte duyarlılık ifade eden parametreler parantez içinde yazılır. Verilen parametrelerde değişim olması durumunda, program yukarıdan aşağıya doğru gerçekleştirilir. Bir mimaride tek 'process' olmak zorunluluğu yoktur. Çoklu yapılarda tüm 'process' ler aynı anda gerçeklenir ancak 'process' içindeki işlemler yukarıdan aşağı doğru yapılır. Böylelikle aynı program içerisinde bağımsız bloklar oluşturulabilir

2.4.1.2. Veri akışı mimarisi

Veri akışı mimarisinde devre tasarımı; karşılaştırıcılar, toplayıcılar, kod çözücüler ve basit lojik kapılar kullanılarak tasarlanır. Program içerisindeki satırlar tamamen eşzamanlı olarak çalışır. Davranışsal mimariden farklı olarak duyarlılık ifade eden parametreler, program satırlarındaki eşitliklerin sağ tarafında kalan parametrelerdir.

Bu tür mimaride davranışsal mimariye göre dışarıdan bakıldığında yapılacak işlemin anlaşılırlığı daha azdır. Aynı zamanda yapılacak işlemin gücü büyüdükçe her fonksiyonu standart elemanlarla ifade etme zorluğu nedeniyle, genelde kullanıcı davranışsal mimariyi tercih eder. Ancak davranışsal mimariye göre, devrenin kaplayacağı alan bakımından daha kontrollüdür.

2.4.1.3. Yapısal mimari

Yapısal mimari temel olarak tamamen kullanıcının kontrolündeki bir tasarım biçimidir. Bu tasarımda tüm bağlantılar yazılımcı tarafından tanımlanır ve işaret isimleri verilerek belirlenir. Ayrıca kullanılacak elemanlar yazılımcı tarafından 'component' olarak oluşturulur ve tasarımda kullanılır.

Yapısal mimari donanım tasarımında yapılacak çizimin, yazılım ile gerçeklenmesi olarak görülebilir. Bu nedenle sistem karmaşıklığı arttıkça bu tasarımın kullanımı çok zorlaşacaktır. Küçük projelerde ya da genelde aynı yapıların tekrar ettiği sistemlerde kullanılabilir. En büyük avantajı tasarımın kaplayacağı alanın kullanıcı kontrolü altında olmasıdır. Bu tez çalışmasında Quartus II ile gerçekleştirilen tasarım yapısal mimariye uygun olarak tasarlanmıştır.

2.5. FPGA Modellemesinde Kayan Noktalı (Floating Point) Sayılar

Kayan noktalı sayılar gerçel sayıların bilgisayar ortamındaki gösterim şekillerinden biridir. Gerçek dünyada sayılar sonsuza kadar giderken, bilgisayar ortamında bilgisayar donanımının getirdiği sınırlamalardan dolayı bütün sayıların gösterilmesi mümkün değildir. Bununla birlikte gerçekte sonsuza kadar giden birtakım değerler bilgisayar ortamında ortamın kapasitesine bağlı olarak yaklaşık değerlerle temsil edilirler. Bu sınırlamaların etkisini en aza indiren, sayıların maksimum miktarda ve gerçeğe en yakın şekilde temsilini sağlayan sisteme "Kayan-Noktalı Sayılar" sistemi denir. Kayan noktalı sayılar sistemi, bir sayı ile 10'un herhangi bir kuvvetinin çarpımı şeklinde sıklıkla kullanılan bilimsel gösterime oldukça benzeyen bir notasyona sahiptir ve en sık kullanılan IEEE 754 standardına göre şekillendirilmiştir.

Bilinen gösterim şekillerinde n bitlik kapasiteyle gösterilebilecek sayı aralığı bellidir. İşaretsiz gösterimde; 0 ile 2n, bire tümleyen şeklindeki gösterimde; $-2^{n-1}+1$ ile 2^{n-1}, ikiye tümleyen gösteriminde ise; -2^{n-1} ile 2^{n-1} arasındaki sayıları göstermek mümkündür.

Bilimsel gösterimde sayılar, virgülün solunda 0'dan farklı bir basamak kalacak şekilde 10'un kuvvetiyle çarpım halinde gösterilir. Örneğin; 5647 ve 0.0003456 sayıları sırasıyla 5.647×10^{3} ve 3.456×10^{-4} şeklinde yazılır. Bu formatta 10'un kuvveti değiştikçe çarpan sayının tam sayı kısmını belirleyen nokta, kuvvetin değişme yönüne uygun olarak

kaydırılır. Bu kaydırma işlemine olağanlaştıma (normalizasyon) denir. Kayan noktalı sayılar sistemi de bu temel prensibe dayalı bir sisteme sahiptir.

Kayan noktalı sayılar, ikilik düzendeki sayıların bilimsel gösterimle gösterilmesidir. Kayan noktalı sayılar işaret, anlamlı kısım ve üst (2'nin üssü şeklinde) olmak üzere üç kısımdan oluşur.

M x BE

Burada M kayan noktalı sayının mantis'i; B tabanı, E ise üssüdür. Kayan noktalı sayının gösteriminde üst bitleri fazla olursa sayının duyarlılığı, anlamlı kısmın bitleri fazla olursa gösterilebilecek sayı aralığı artar.

Kayan noktalı sayıların her değişik aygıta uyumlu bir gösterime sahip olması için IEEE-754 standardı geliştirilmiştir. Bu standarda göre kayan noktalı sayıların gösterimi tek duyarlı(32 bit) ve çift duyarlı(64 bit) olmak üzere iki şekilde belirlenmiştir. Bu gösterimlerde sayılar olağanlaştırılmış ve üstler, üst değerini tutan bit sayısına göre saptırılmıştır. Üst değerini tutan bit sayısı k ise, saptırma değeri 2^{k-1}-1 olarak belirlenmiştir. Bu tez çalışmasında 32 bit IEEE-754 kayan noktalı sayı standardı kullanılmıştır.

2.5.1. Olağanlaştırma (Normalizasyon)

Kayan noktalı sayıları bu yöntemle ifade ederken karşılaşılan sorunlardan biri de sayıların benzersiz gösteriminin sağlanamamasıdır. Kayan nokta yöntemiyle gösterim yapılırken virgülün yeri ile birlikte üst değeri değiştiğinde aynı sayı birden farklı şekilde yazılabilir. Sayıların tümü aynı değere sahiptir, ancak gösterim farkından dolayı değişik şekillerde ifade edilebilmektedirler. Bu durum bilgisayarlar için verimli olmadığından sayıların eşsiz gösterimini sağlayan bir yöntem hayata geçirilmiştir. Bu yöntemde ikilik sistemdeki sayılar bilimsel gösterimdeki gibi, virgülün solunda sıfırdan farklı bir sayı kalana kadar kaydırılır ve o şekliyle ifade edilir. Bu işleme olağanlaştırma (normalizasyon) denir. Bu işlem sonunda her sayının benzersiz gösterimi sağlanırken, sayı 1 ve 0'lardan oluştuğu için virgülün solunda sıfırdan farkı olan değerin 1 olduğu bilindiğinden, anlamlı kısmın ifadesi için fazladan bir bit kazanılmış olur.

Örneğin: 11110000 11001100 10101010 00000000

+1.1110000 11001100 10101010 00000000

Şeklinde gösterilir.

2.5.2. IEEE-754 Formatına Dönüşüm

Bu tez çalışmasında kayan noktalı sayı işlemlerinde 32 bitlik format kullanılmıştır. Aşağıda örnek bir format dönüşümü gösterilmiştir. Örneğin onluk tabandaki "-2345.125_{10}" sayısının 32 bitlik IEEE-754 kayan noktalı sayı (floating point) sayı formatına dönüşümü aşağıda gerçekleştirilmiştir. Öncelikle onluk tabandaki sayı ikilik tabana çevrilir. Bu durumda

$-2345{,}125_{10} = -100100101001.001_2$

İkilik tabana çevrilen sayı 1.M formun yani olağanlaştırma (Normalizasyon) formuna dönüştürülür. Böylece

$-1.00100101001.001 \times 2^{11}$ elde edilir.

Sayı negatif olduğu için IEEE-754 formatındaki işaret biti S=1 olarak belirlenir. Üsse bias değeri(127) eklenerek 8 bitlik E (üst=exponent) hesaplanır. E=e+127 olduğundan $E=11+127=138_{10}=10001010_2$ olur.

M (mantisa) için yukarıdaki 1.M ifadesinden M alınır ve 23 bitlik M=00100101001001000000000 elde edilir.

Hesaplan S,E ve M değerleri birleştirilerek sayının IEEE-754 sayı formatındaki 32 bitlik sayı elde edilir.

1	10001010	00100101001001000000000
S	E	M

EK-2'de kayan noktalı sayılar üzerinden dört işlemin nasıl gerçekleştirildiği ayrıntılı olarak verilmiştir.

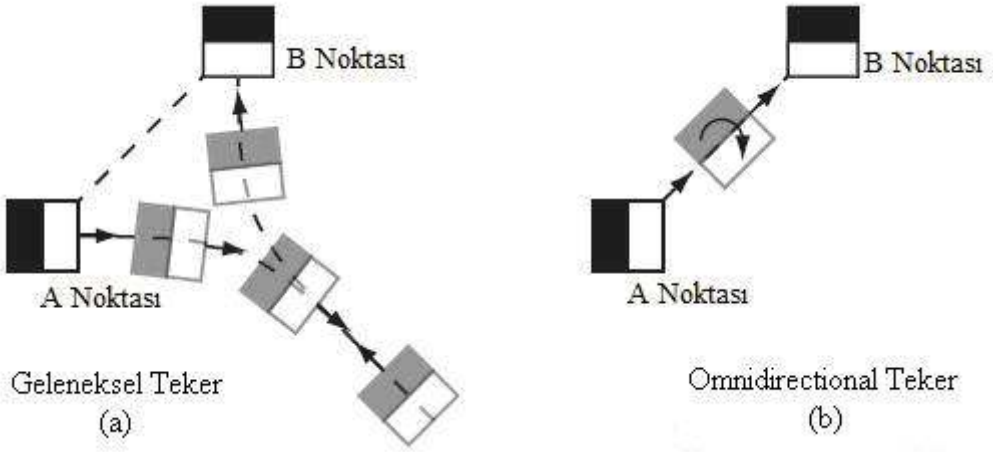
3. OMNI DIRECTIONAL MOBİL ROBOTLAR

Mobil robotların endüstriyel ve teknik uygulamalardaki önemi sürekli artmaktadır. Genelde robotlar insanların ihtiyaçlarını karşılayan ve insanların yaptıkları işleri daha verimli bir şekilde yerine getiren akıllı makineler olarak ta tanımlanabilir. Mobil robotlar taşıma, kontrol, gözetleme gibi oldukça geniş kullanım alanlarına sahiptirler [21]. Bir özerk mobil robot operasyon bölgesinde engellerden kaçınarak en kısa yoldan işlevi yerine getirmek gibi temel işlemleri yapabilme kabiliyetine sahip olmalıdır. Bu robotların performansını geniş bir alanda izleme, lokalizasyon ve algılama yetenekleri belirler. Robotlar kullanıldıkları çalışma ortamlarına göre ya da belirli bir işi yapabilecek kapasitede özel amaçlı olarak tasarlanabilirler [6,14]. Omni directional mobil robotların en önemli özellikleri, robotun tekerleklerinin birbirlerinden bağımsız olarak hareket edebilmeleridir. Bu da robot mekanizmasının serbestlik derecesinin arttırılması demektir.

3.1 Omni Directional Hareket

Robotik tekerlekler daha çok düzlemsel hareketler için tasarlanmışlardır. İki boyutlu düzlemde belirlenen serbestlik derecesi (nDOF) ile hareket edilir. Ağırlık merkezine göre (ϕ) açısıyla x ve y koordinatlarında istenilen yöne hareket edilebilir. Geleneksel robot tekerleklerinde hareket ve manevra kabiliyetleri sınırlıdır her yöne hareket edemeyen bu robotlara "non-holonomic" robotlar denir. Fakat Omni directional robotlar yüksek hareket kabiliyetine sahiptir. Bu robotlar, sürüş eksenine dikey yönde çok rahat bir şekilde hareket edebilirler [19].

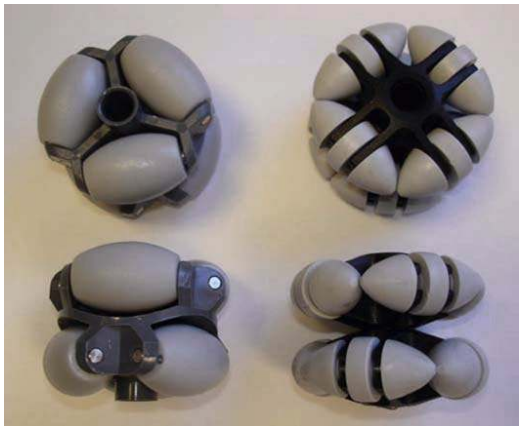
Omni directional mobil robot tekerleklerinin açısal doğrultuları değiştirilmeden sadece tekerlerin dönme doğrultuları değiştirilerek robotun istenilen yönde hareketini sağlanabilir. Oysaki geleneksel guruptaki robotların istenilen yönde hareketini sağlamak için en az iki adet tekerleğin yönlendirilebilmesi yani ayrı bir serbestlik derecesi gerektirmektedir. Bu da hem denetimi zorlaştırmakta hem de maliyeti arttırmaktadır [17]. Omni directional hareketin birçok çeşidi geliştirilmiştir. Özellikle statik ve dinamik engelleri aşmada bu hareketlilik mükemmel sonuçlar sağlar [8,16]. Şekil 3.1 (b) 'de omni directional mobil robot A noktasından B noktasına giderken kendi üzerinde de 90^o derece dönmektedir. Bu hareket geleneksel robotta şekil 3.1 (a)'daki gibi ancak iki aşamada gerçekleşmektedir.



Şekil 3.1. Omni directional hareket ve geleneksel hareket

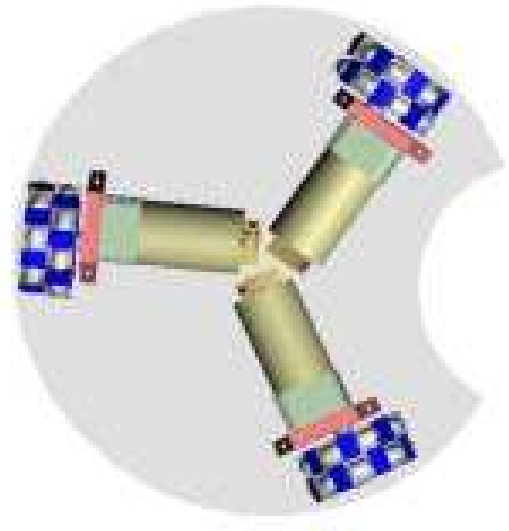
Omni directional hareketi sağlayan omniwheel, macnum, ortagonal gibi birden çok tekerlek yapısı vardır. Bunlardan en etkin olarak kullanılan macnum tekerlek yapısıdır. Macnum tekerleklerin yüzeyi birçok serbest dönen küçük silindirlerle (rollers) kaplıdır. Tekerlek merkezinin bir motor tarafından hareket ettirilirken, tekerlek yüzeyindeki küçük silindirlerin (rollers) bu motor tarafından kontrol edilmemesi önemli bir yapı oluşturur. Bu küçük silindirler, silindir yatakları sayesinde bir yerde tutulur ve kendi eksenlerinde serbestçe dönebilirler. Küçük silindirlerde (roller) güç, silindir eksenine paralel ve dik vektörlere bölünebilir. Silindirlerin dikey eksenine verilen güç silindirlere küçük bir dönüş hareketi sağlarken, silindir eksenine paralel olan güç ise tekerleğe ve dolayısıyla robota güç sağlar.

Küçük silindir tekerlekler mobil robot tekerleği üzerine, tekerleğin dönme eksenine göre farklı açılarda yerleştirilerek istenilen amaca yönelik tekerleğe farklı tiplerde çok yönlülük kazandırılmaktadır. Küçük silindir tekerleklerin, robot tekerleğinin dönme eksenine göre yerleştirildikleri açılara bağlı olarak isim verilmektedir. Küçük silindirik tekerlek robot tekerleğinin dönme eksenine göre oluşturduğu açı 90° ise buna Macnum 90°'lik tekerlek, küçük silindirik rulolu tekerlek robot tekerleğinin dönme eksenine göre oluşturduğu açı 45° ise buna macnum 45°'lik tekerlek adı verilmektedir. Şekil 3.2'de biri birine 90° lik açılarla yerleştirilmiş küçük silindirlerin yer aldığı macnum tekerlek yapısı gösterilmiştir. [4,5].



Şekil 3.2. Macnum tekerlek yapısı

Omni directional robotlarda kullanılacak tekerlek sayısı ihtiyaçlara göre belirlenir. İki tekerlekli omni directional robot yani iki serbestlik dereceli robotlar hedef noktaya sadece ileri ve geri hareket ederek ya da dönerek hareket edebilirler. Fakat bu robotlar hedef noktaya yan doğrultuda yaklaşarak hareket edemezler. Üç serbestlik dereceli robotların kontrolü ve idaresi basittir, fakat bu robotlar sınırlı çekiş gücüne sahiptir. Dört serbestlik derecesine sahip robotlarda robotun çekiş gücü yüksektir fakat bu beraberinde robotun mekanik ve kontrolünde karmaşıklık getirir. Bu çalışmada üç serbestlik derecesine sahip üç tekerlekli omni directional robot kullanılmıştır. Şekil 3.3'de tekerlekleri arasında 120^{o}'lik açı farkı olan 3 tekerlekli omni directional mobil robot gösterilmiştir [22,23].

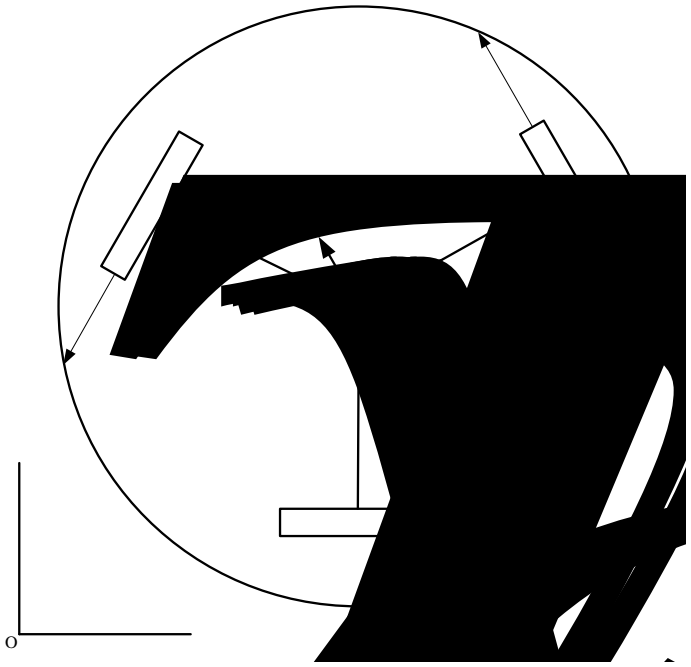


Şekil 3.3. 3 Serbestlik derecesine sahip omni directional robot

3.2 Robot Platformunun Kinematiği ve Ters Kinematiği

Robotun sabit eksene göre genel hızıyla, robotun tekerlek merkezlerinin dönme hızları arasındaki ilişki kinematik denklemler ile kurulur. Kinematik eşitliğin bilinmesi durumunda ters kinematik kolayca bulunabilir. Ters kinematik eşitliği gerekli kartezyen hız girişini alır ve onu her bir tekerlek için açısal hızlara dönüştürür [24].

Robotu temsil eden koordinatların platformun merkezine yerleştirildiği düşünülmektedir. Şekil 3.4 'de robotun koordinat sistemi gösterilmektedir



Şekil 3.4. Robot için Koordinat sistemi

Şekil 3.4'de görüldüğü üzere $x_w o y_w$ sabit eksen koordinatları, $x_m o y_m$ ise hareketli eksen koordinatlarını ifade etmektedir. Hareketli eksen koordinatları, sabit eksen koordinatları ile aynı orijin noktasına sahiptir fakat hareketli eksen koordinatları robot ile birlikte dönmektedir. x_m ekseni, T_1 çekme kuvvetine göre dik doğrultuda yer almaktadır ve ϕ hareketli eksenle sabit eksen arasındaki açıyı temsil etmektedir [6].

Hareketli eksendeki kuvvetler arasındaki ilişki geometriksel olarak aşağıdaki şekilde yazılabilir:

$$F_{mx} = \sqrt{\frac{3}{2}}\, T_2 - \sqrt{\frac{3}{2}}\, T_3 \tag{3.1}$$

$$F_{my} = T_1 - \frac{1}{2} T_2 - \frac{1}{2} T_3 \tag{3.2}$$

$$T_z = LT_1 + LT_2 + LT_3 \tag{3.3}$$

Şekil 3.4'de gösterilmekte olan *L*, robot ağırlık merkezi ile bir tekerlek arasındaki mesafedir. ${}^{m}F = \begin{bmatrix} F_{mx} & F_{my} & T_z \end{bmatrix}^T$ hareketli eksendeki robotun x ve y yönündeki kuvvetlerini ve robotun toplam momentini belirtmektedir.

${}^{t}F = [T_1 \;\; T_2 \;\; T_3]^T$ tekerleklerin itme kuvvetlerini göstermektedir.

Burada eşitlik (3.1-3.2) matris şeklinde düzenlenirse aşağıda verilen eşitlik 3.4 elde edilir. *B* katsayı matrisi olup eşitlik 3.5'de verilmiştir.

$${}^{m}F = B\, {}^{t}F \tag{3.4}$$

$$B = \begin{bmatrix} 0 & \sqrt{\frac{3}{2}} & -\sqrt{\frac{3}{2}} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ L & L & L \end{bmatrix} \tag{3.5}$$

Enerjinin korunumu prensibinden faydalanılarak yukarıdaki eşitliklerden eşitlik 3.6 elde edilebilir:

$${}^{W}F^T \cdot {}^{W}\dot{X} = {}^{m}F^T \cdot {}^{m}\dot{X} = {}^{t}F^T \cdot {}^{t}\dot{X} = {}^{t}F^T \cdot r\dot{q} \tag{3.6}$$

Burada ${}^{w}F = [F_x \;\; F_y \;\; T_z]^{\,T}$ sabit eksen sistemindeki kuvvetlerdir. ${}^{W}\dot{X} = [\dot{x}_w \;\; \dot{y}_w \;\; \dot{\omega}_w]^T$; mobil robotun sabit eksene göre hızını, ${}^{m}\dot{X} = [\dot{x}_m \;\; \dot{y}_m \;\; \dot{\omega}_m]^T$; mobil robotun hareketli eksene göre hızını göstermektedir. r; tekerleklerin yarıçapı ve $\dot{q}$; tekerleklerin açısal hızları, $\dot{\omega}$; robotun açısal hızıdır.

Bir tekerin hızı; ${}^{W}\dot{X} = r\dot{q}$ olarak ifade edilir.

Şimdi robotun merkezine iliştirilmiş hareketli eksenin hızı şu şekilde yazılabilir:

$${}^{m}\dot{X} = (B^T)^{-1}\, r\dot{q} \tag{3.7}$$

Bu hızın sabit eksen sistemindeki karşılığı bilinmek zorunda olduğu için, bir dönüşüm matrisi kullanılmak zorundadır. Bu dönüşüm matrisi koordinatların, hareketli eksenden sabit eksene dönüştürülmesine olanak sağlamaktadır. Dönüşüm matrisi şu şekilde verilir.

$$ {}^{W}_{m}R = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8} $$

Burada ϕ Şekil 3.4'de gösterilmekte olan hareketli eksenle sabit eksen arasındaki açıyı temsil etmektedir.

Yukarıdaki dönüşüm matrisi sayesinde hareketli eksenin hız ve ivmesi sabit eksene göre düzenlenebilir. Eşitlikler aşağıda verilmektedir.

$$ {}^{W}\dot{X} = {}^{W}_{m}R \, {}^{m}\dot{X} = {}^{W}_{m}R(B^{T})^{-1} \, r\dot{q} \tag{3.9} $$

$$ {}^{W}\ddot{X} = {}^{W}_{m}\dot{R}(B^{T})^{-1} \, r\dot{q} + {}^{W}_{m}\dot{R}(B^{T})^{-1} \, r\ddot{q} \tag{3.10} $$

Eşitlik 3.9'un açık şekilde yazılırsa eşitlik 3.11 elde edilir.

$$ \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = r \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & L \\ \sqrt{\frac{3}{2}} & -\frac{1}{2} & L \\ -\sqrt{\frac{3}{2}} & -\frac{1}{2} & L \end{bmatrix}^{-1} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{3.11} $$

Eşitlik 3.9 robotun gerçek pozisyonunu bulmak için kullanılacak olan ileri kinematik eşitliğidir. Bu eşitlik kodlayıcılar tarafından hesaplanmış olan gerçek açısal hızları alacak ve onları robotun gerçek hızlarına dönüştürecektir. Bu hızlar daha sonra robotun gerçek pozisyonunu bulmak için kullanılabilir. Yani bir sabit eksen görüş sistemi kullanılmaksızın robotun pozisyonunun nasıl bulunacağını ifade etmektedir. Buradaki dezavantaj kaymanın hesaba katılmamasıdır.

Kinematik eşitliğin bilinmesi durumunda ters kinematik kolayca bulunabilir. Ters kinematik eşitliği kullanıcıdan gerekli Kartezyen hız girişini alır ve onu her bir tekerlek için açısal hızlara dönüştürür. Bu denklem eşitlik 3.12'de verilmiştir [6,7].

$$\dot{q} = B^T {}^W_m R^{-1}({}^W\dot{X})/\mathrm{r}$$

(3.12)

Eşitlik 3.12 açık şekilde yazılırsa eşitlik 3.13 elde edilir.

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = \frac{1}{r}\begin{bmatrix} 0 & 1 & L\\ \sqrt{\frac{3}{2}} & -\frac{1}{2} & L\\ -\sqrt{\frac{3}{2}} & -\frac{1}{2} & L\end{bmatrix}\begin{bmatrix}\cos(\phi) & -\sin(\phi) & 0\\ \sin(\phi) & \cos(\phi) & 0\\ 0 & 0 & 1\end{bmatrix}^{-1}\begin{bmatrix}V_x\\ V_y\\ \omega\end{bmatrix}$$

(3.13)

3.3 Robotun Dinamik Modeli

Robotun hem sayısal hem de donanımsal simülasyonları çalışırken uygun şekilde temsil edilebilmesi için sistemin dinamik modelinin çıkartılmasına ihtiyaç duyulur. Robot için dinamik model çıkarma Newton'un ikinci kuralı kullanılarak başlar.

$${}^W F = M\,{}^W\ddot{X} = {}^W_m RB\ {}^t F$$

(3.14)

Burada ${}^W\ddot{X}$ matrisi, robotun X ve Y yönündeki ivmelerini ve ayrıca robotun kendi merkezi etrafında yaptığı dönme hareketinin ivmesini ihtiva eder. Bağıntıda verilen tüm hareket ve koordinatlar sabit eksene aittir. M robotun ağırlık matrisidir.

$$M = \begin{bmatrix} m & 0 & 0\\ 0 & m & 0\\ 0 & 0 & J\end{bmatrix}$$

(3.15)

Ağırlık matrisindeki m kütle, J ise robotun kütlesel atalet momenti.

$$({}^W_m R)^{-1} = {}^W_m R^{\,T}$$

Eşitlik 3.10, 3.14 ve yukarıda verilmiş olan dönüşüm matrisinin ortogonal özelliği kullanılarak robotun çekme kuvveti şu şekilde elde edilir:

$$^{t}F = (B)^{-1}{}_{m}^{W}R\,^{T}M[{}_{m}^{W}R(B^{T})^{-1}\,r\dot{q}_L + {}_{m}^{W}R(B^{T})^{-1}\,r\ddot{q}_L]$$

(3.16)

Şimdi yük momenti (tork) olan τ_L bulunur ve buradan da motor momenti olan τ_M hesaplanabilir.

$$\tau_L = J_L\ddot{q}_L + c_L\dot{q}_L + {}^{t}Fr \tag{3.17}$$

$$\tau_m = J_m\ddot{q}_m + c_m\dot{q}_m + (1/n)\tau_L \tag{3.18}$$

$$\dot{q}_L = (1/n)\dot{q}_m \tag{3.19}$$

Burada $\dot{q}_m = [\dot{q}_{m1} \quad \dot{q}_{m2} \quad \dot{q}_{m3}]^T$ matrisi üç motorun açısal hızlarını içerir. τ moment (tork), c sönümleme katsayısı (damping) ve J ise kütlesel atalet momentini ifade eder. Bu simgelerin alt simgeleri olan $'L'$ ve m' sırasıyla yük (tekerlek) ve motoru temsil etmektedir. n, motorun redüktör dişli oranını ifade eder. Bu dişli transmisyon düzlemi tekerleklerin açısal hızını düşürür fakat tekerleklerin momentini artırır.

Yukarda tanımlanan eşitlikler eşitlik 3.20'de matris formuna çevrilerek robotun tüm simülasyonlarında kullanılan nihai moment (tork) eşitliği elde edilmiş olur.

$$\tau_M = k_1\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\ddot{q}_m + k_2\begin{bmatrix}0&1&1\\1&0&1\\1&1&0\end{bmatrix}\ddot{q}_m + k_3\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\dot{q}_m + k_4\dot{\phi}\begin{bmatrix}0&1&-1\\-1&0&1\\1&-1&0\end{bmatrix}\dot{q}_m$$

(3.20)

Burada,

$$k_1 = J_m + \frac{J_L}{n^2} + \frac{(4mL^2+J)r^2}{9L^2n^2} \tag{3.21}$$

$$k_2 = \frac{(-2mL^2+J)r^2}{9L^2n^2} \tag{3.22}$$

$$k_3 = c_m + \frac{c_L}{n^2} \tag{3.23}$$

$$k_4 = \frac{2\sqrt{3}\,mr^2}{9n^2} \tag{3.24}$$

$$\dot{\phi} = \frac{r(\dot{q}_1+\dot{q}_2+\dot{q}_3)}{3L} = \frac{r(\dot{q}_{m1}+\dot{q}_{m2}+\dot{q}_{m3})}{3L} \tag{3.25}$$

Dinamik model parametrelerinin hesaplanışı EK-4'te ayrıntılı verilmiştir.
Bir motor için aşağıdaki denklem verilebilir.

$$\tau_m = (E - k_E \dot{q}_m) + k_{lr} k_M$$

(3.26)

Burada E motor giriş gerilimi, k_E motorun zıt EMK sabiti ve k_M motor moment sabitidir. k_{lr}, $1/R$ ifadesine eşittir ki burada R motorun armatürüdür. Motorun indüktansı, küçük olduğu için ve genellikle robot dinamiklerinde önemsenmediği için ihmal edilmiştir.

Üç tekerlekli omni directional robotun birleşmiş doğrusal olmayan hareket denkleminin (EOM; 'Equations of Motion') son hali olan eşitlik 3.27 elde edilir.

$$E = \frac{k_1}{k_{lr} k_M} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ddot{q}_m + \frac{k_2}{k_{lr} k_M} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \ddot{q}_m + \frac{k_3}{k_{lr} k_M} + k_E) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{q}_m + \frac{k_4 \dot{\phi}}{k_{lr} k_M} \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \dot{q}_m$$

(3.27)

Doğrusal olmayan terimin etkisi motorun dönme hızı $\dot{\phi}$ ve $k_4/k_{lr}k_M$ katsayısı ile orantılıdır. Robot kendi üzerinde dönmeden hareket ettiğinde doğrusal olmayan terim sıfır olur [6,9].

4. MOBİL ROBOT MODELİNİN FPGA TASARIMI

Bu tez çalışmasında omni directional mobil robotun belirtilen yörüngeyi izlemesi için tekerlek hızlarının PI denetleyici ile hız değerlerinin istenilen değerlerde olması sağlanmıştır. Bunun için Matlab'da oluşturulan model Altera firmasının Stratix II GX serili FPGA'sı için Quartus yazılım arayüzü kullanılarak FPGA ortamında tasarım gerçekleştirilmiştir. Daha sonra elde edilen sonuçlar Matlab'da oluşturulan modelin sonuçları ile karşılaştırılarak doğrulanmıştır. Referans alınan omni directional mobil robot 3 tekerlekli olup tekerleklerin robot ağırlık merkezine uzaklığı 0.06 m, tekerlek yarıçapları 0.05 m ve tekerlekler arasındaki açı 120^{o} dir.

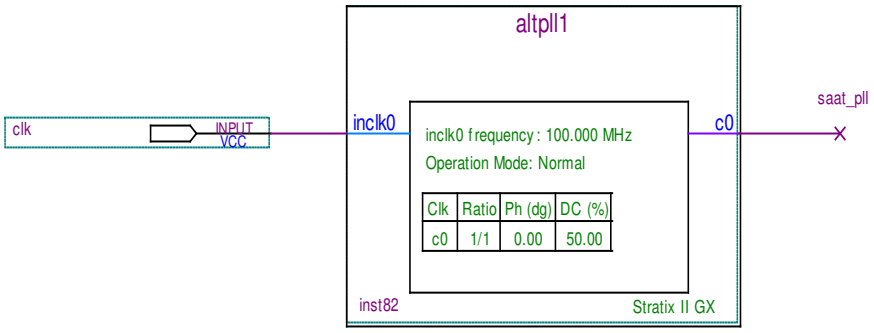
Şekil 4.1'de gösterildiği gibi sistem bloklar halinde tasarlanmıştır. FPGA ortamında modellenen ve gerçek zamanlı çalıştırılacak olan omni directional mobile robotun farklı yörüngelerde hareketini sağlamak için X yönündeki hızı (Vx), Y yönündeki hızı (Vy), robotun açısal hızı (ω) ve mobile robotun yönünü belirlemek için gerekli $\sin\phi$ ve $\cos\phi$ değerleri FPGA geliştirme kartı üzerindeki ROM hafıza birimlerine kaydedilir. Ters kinematik modelde hafıza birimlerinden gelen verilerden her bir tekerleğin ayrı ayrı referans hızları hesaplanır. PI denetleyici modülünde; gerçek hız değerleri referans hız değerleri ile karşılaştırılarak robot tekerleklerinin gerekli hızlarda olması sağlanmaktadır. Dinamik modelde; tekerlek dönüş hızları, tekerlek ivmeleri ve dinamik model parametreleri ile her bir tekerleğe uygulanması gereken moment değerleri hesaplanır. DC motor modelinde tekerleklerin istenilen hızlarda dönmesi için tekerleklere uygulanması gereken moment hesaplanır. Bu tez çalışmasında oluşturulan sistem için Quartus II 8.1 versiyonu kullanılarak Altera firmasının Stratix II GX EP2SGX90FF1508C3 FPGA'sı için tasarım gerçekleştirilmiştir. Model hesaplamalarında 32 bit IEEE-754 kayan noktalı sayı standardı kullanılmıştır.

Şekil 4.1'de mobil robot modeli için gerçekleştirilen FPGA tasarımının blok yapısı gösterilmiştir.

Şekil 4.1. FPGA tasarımının blok yapısı

4.1. Clock Üretimi ve Özel Sinyallerin Oluşturulması

Şekil 4.2 ‘de gösterilen alt devre, bir FPGA bloğu olan altpll1 vasıtasıyla girişine uygulan clock işaretinden FPGA’ın her noktasında eş zamanlı olarak kullanılabilecek bir clock işareti üretir ve bu işareti yine herhangi bir kayma ya da gecikme olmaksızın her noktaya dağıtmayı sağlar. Tasarımda simülasyona başlamadan önce clk girişi 100 MHz’lik %50 görev periyotlu bir kare dalga olarak belirlenir. Çıkıştan alınan saat_pll yine aynı frekans ve görev periyodundaki bir kare dalgadır ve blokların genel clock girişi olarak kullanılır.

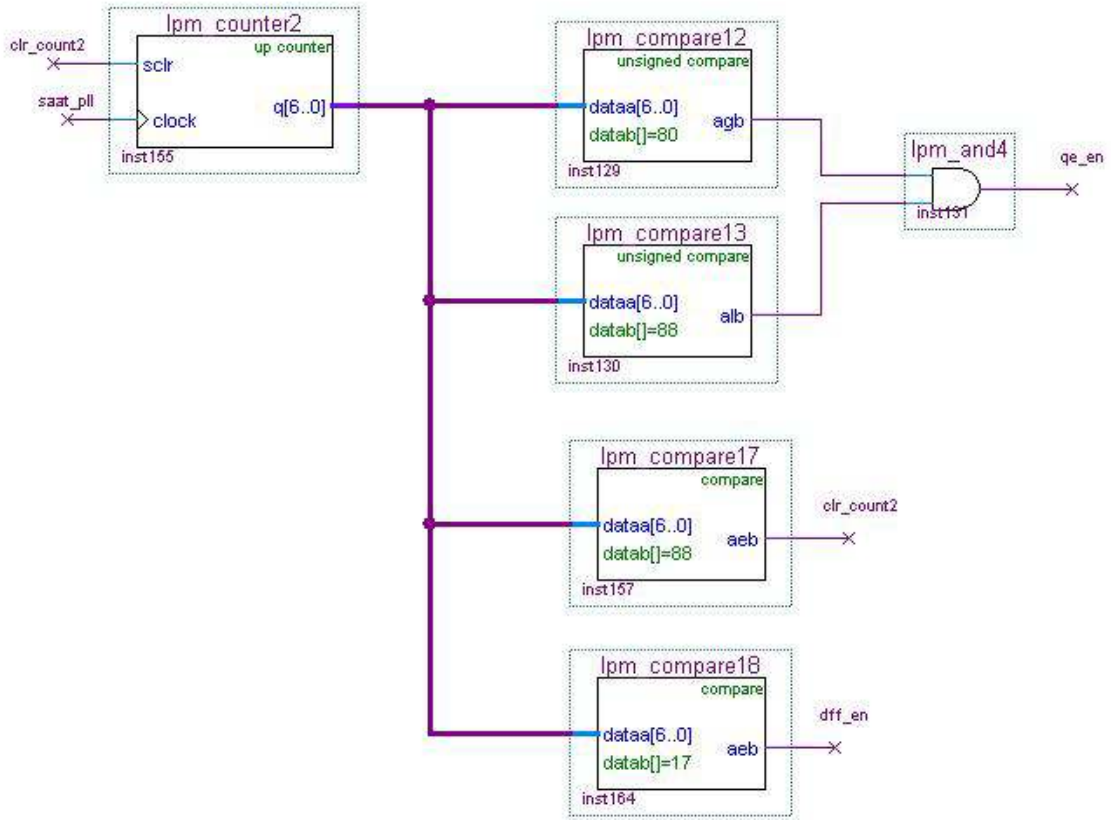


Şekil 4.2. FPGA tasarımı clock kaynağı

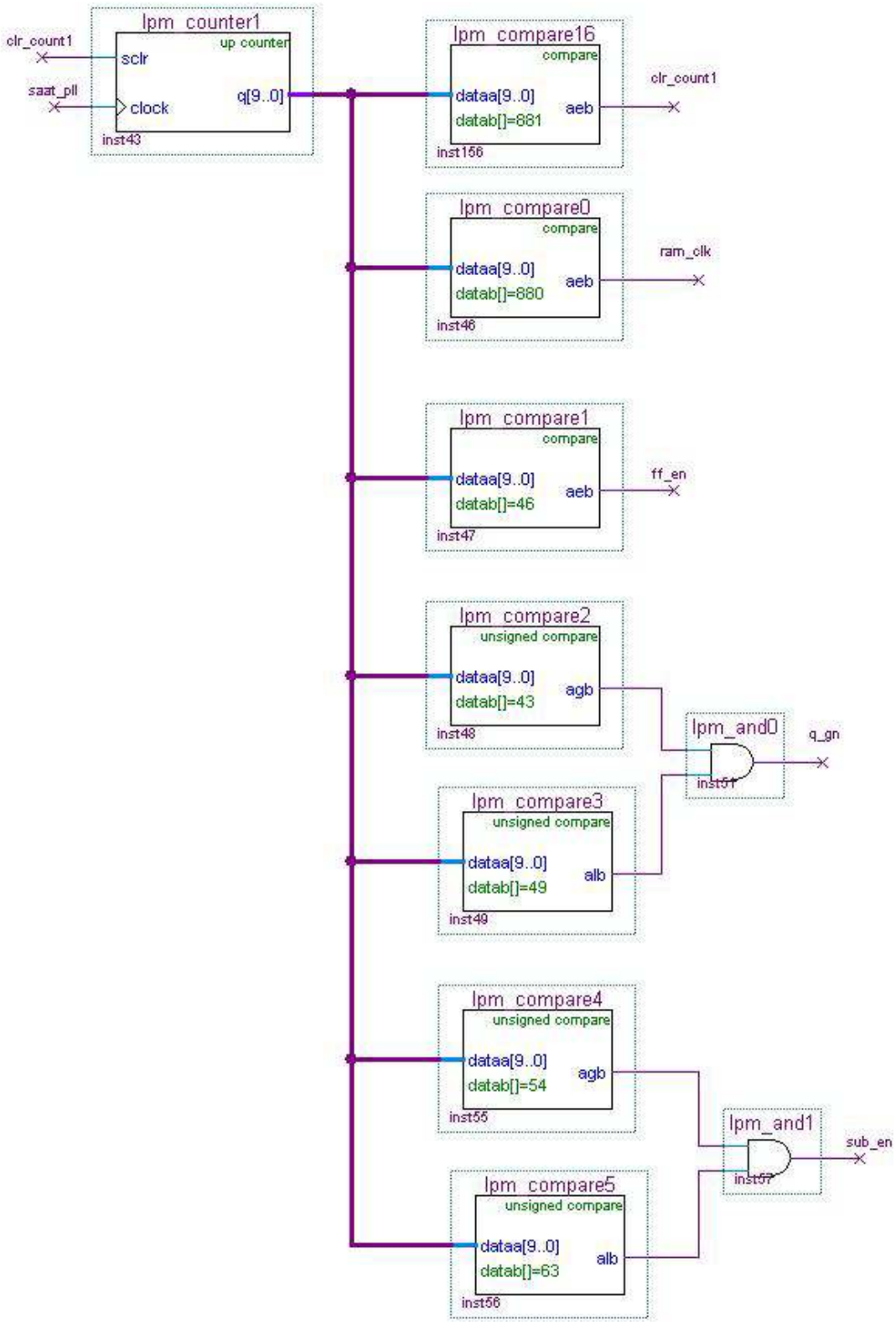
PLL (faz kilitlemeli döngü); bir faz dedektörü, bir alçak geçiren filtre ve gerilim kontrollü osilatörden oluşan elektronik bir devredir. Giriş sinyal frekansı ile VCO (**voltage-controlled oscillator**)’dan karşılaştırma devresine gelen frekans aynı olduğu zaman çıkış olarak alınan Vd gerilimi, VCO’yu giriş sinyali ile kilitli tutmak için gereken değerdir.Ardından VCO, giriş frekansında sabit genlikli kare dalga sinyali üretir. En iyi çalışma, VCO merkez frekansının, kendi doğrusal çalışma aralığının ortasındaki dc öngerilim noktasına ayarlanmasıyla elde edilir. Yükselteç, filtre devresinin çıkışı olarak elde edilen dc geriliminin ayarlanmasını mümkün kılar. Döngü kilitli olduğu zaman, karşılaştırıcıya uygulanan iki sinyal, aynı fazda olmasa da aynı frekanstadır. Karşılaştırıcıya uygulanan iki sinyal arasındaki sabit faz farkı, VCO için sabit bir dc gerilimi oluşturur. Bu durumda giriş sinyali frekansındaki değişmeler, VCO’ya uygulanan dc geriliminin değişmesine neden olur. Yakalama ve kilitleme frekans aralığında dc gerilimi, VCO frekansını sürerek giriş frekansıyla eşitlenmesini sağlar.

Bu proje çalışmasında oluşturulan blokların yanlış ara değer hesaplamalarının önüne geçmek, farklı frekans ve görev periyotlu çalışan blokların gereksinim duyduğu

clock işaretlerini üretmek için özel sinyaller oluşturulmuştur. Oluşturulan sistemdeki pll çıkışındaki clock işaretinden sayıcı ve karşılaştırıcılar kullanılarak blokların gereksinim duyduğu özel işaretler elde edilir. Üretilen çıkış işaretleri alt blokların clock ve enable girişlerine uygulanmaktadır. Örneğin Şekil 4.4'de lpm_compare0 bloğunun ram_clock çıkışı ile her 880 clockta hafızadan veri alınarak veri yoluna iletilmesi sağlanmıştır, lpm_compare16 bloğunun clr_count çıkışı ile 880 clock sonra sistemin yeni veriyi alması için lpm_counter2'nin resetlenmesi sağlanmıştır. Diğer özel işaretler Şekil 4.3 ve Şekil 4.4'de özel sinyal oluşturma devreleri ile gösterilmiştir.



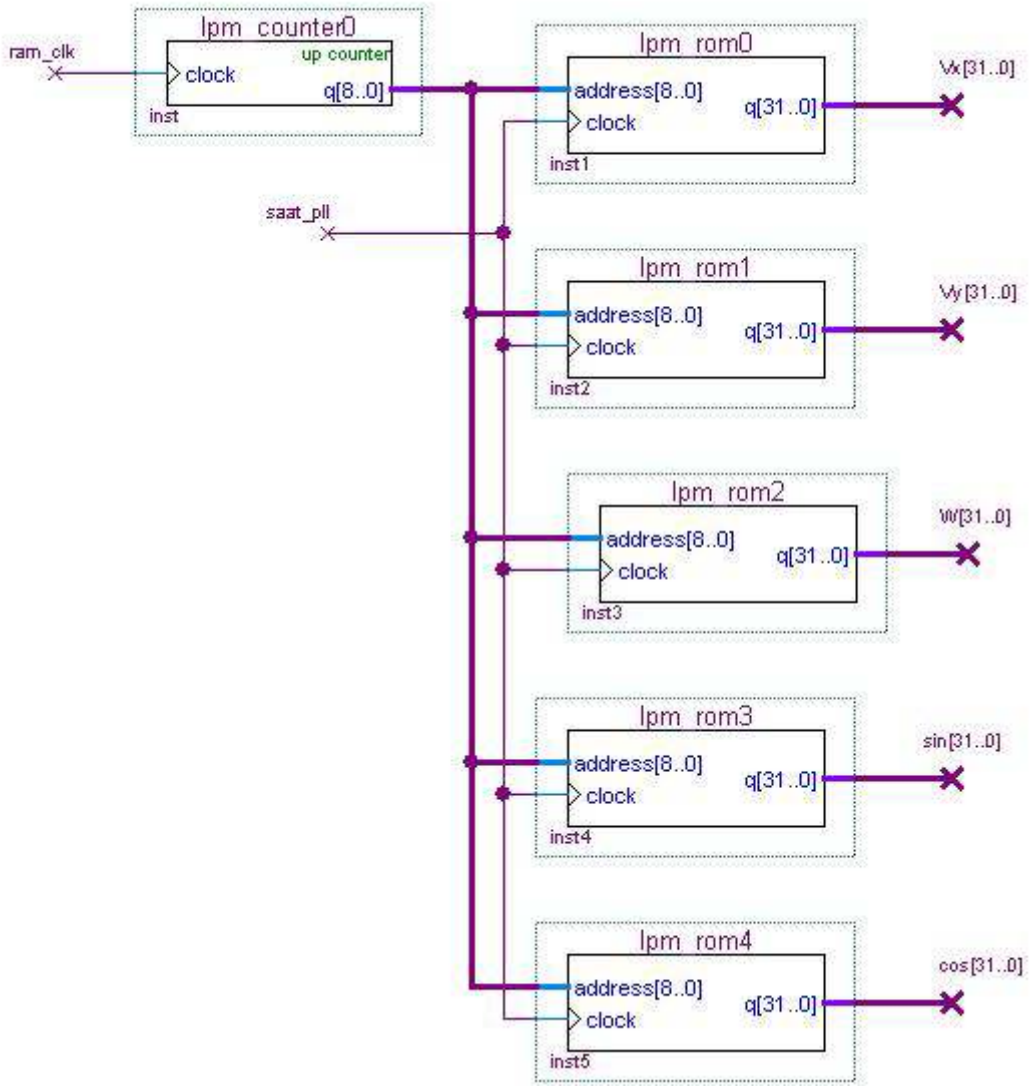
Şekil 4.3 Birinci özel sinyal oluşturma devresi



Şekil 4.4. İkinci özel sinyal oluşturma devresi

4.2. Hafıza Yönetimi

Bu çalışmada oluşturulan mobil robot modelinde robotun belirtilen yörüngede hareketini sağlamak için x ekseni, y ekseni, kendi ekseni etrafındaki açısal hızı ve dönüşüm matrisinden gelen sinϕ ve cosϕ değerleri FPGA üzerinde oluşturulan ROM'lar üzerine kaydedilir. Sistemin her çevrimde bu verilerden biri alınarak sistem veri yoluna verilir. Oluşturulan sitemde bir çevrim süresi 880 clocktur. lpm_counter0 bloğu ile okunacak ROM hücresi adreslenmektedir ve her 880 clockta bir sonraki hafıza hücresini adreslenerek değeri veri yoluna aktarılmaktadır. Şekil 4.5'de oluşturulan ROM hafıza yapısı gösterilmektedir.



Şekil 4.5. Hafıza yönetimi

4.3. Ters Kinematik Model

Robotun ters kinematik modelinin gerçekleştirildiği alt devre, sistem girişleri olan global düzlemdeki x ekseni, y ekseni ve kendi ekseni etrafındaki dönme hızlarını ROM hafıza birimlerinden alarak her bir tekerleğin bağlı olduğu motorların açısal hızlarını hesaplamaktadır.

Mobil robotun bölüm 3.2'de anlatılan ters kinematik eşitliklerinden eşitlik 3.10'nun açılımı sonucu;

$$\dot{q}_1 = (Vy * \cos(a) - Vx * \sin(a) + L*\omega) * n/r$$

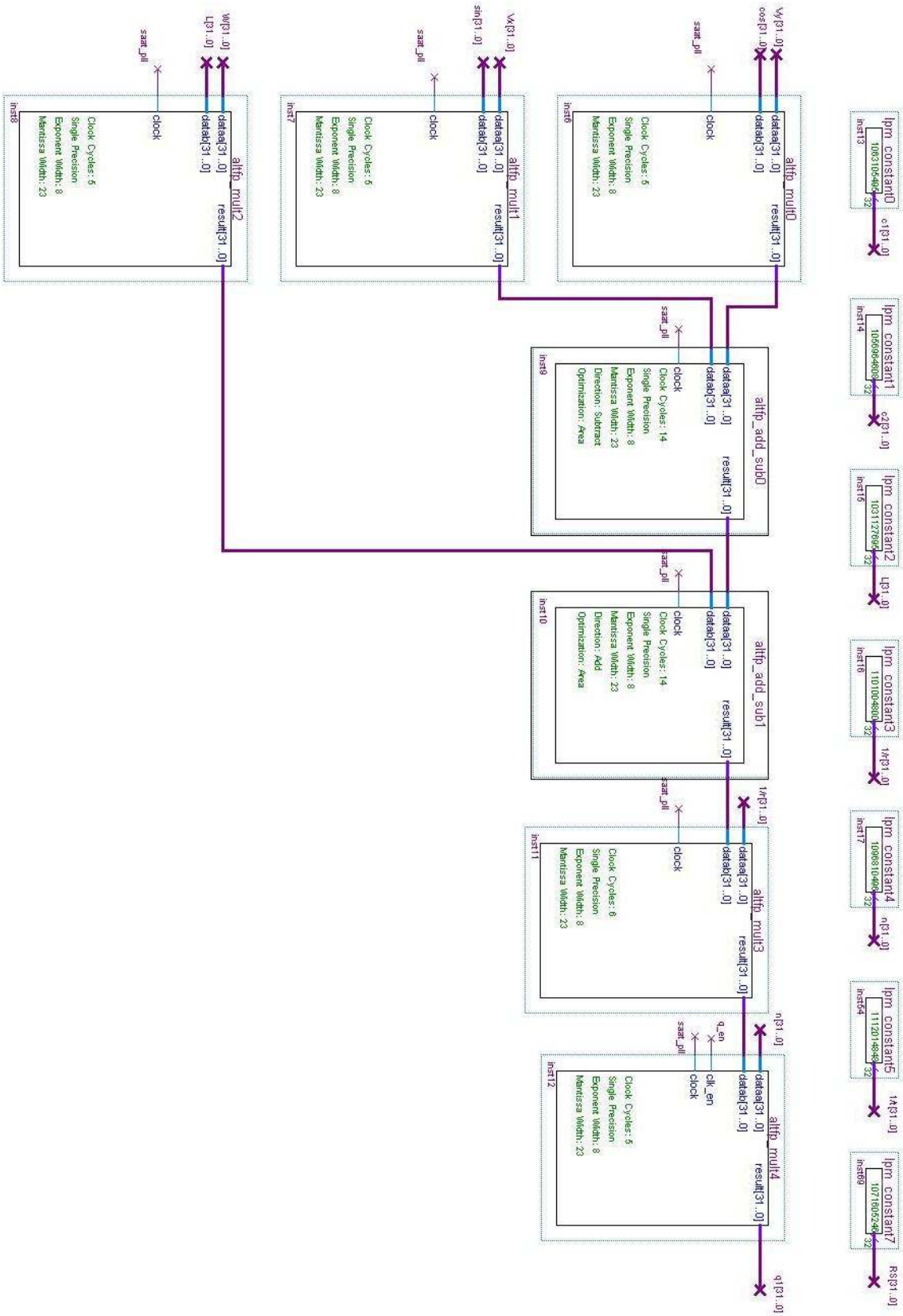
$$\dot{q}_2 = [Vx(c_1*\cos(a) + c_2*\sin(a)) + Vy(c_1*\sin(a) - c_2*\cos(a)) + L*\omega)] * n/r$$

$$\dot{q}_3 = [Vx(c_2*\sin(a) - c_1*\cos(a)) - Vy(c_1*\sin(a) + c_2*\cos(a)) + L*\omega)] * n/r$$

Şeklinde elde edilir.

Burada Vx, robotun x eksenindeki hızını; Vy, robotun y eksenindeki hızını; ω, robotun kendi ekseni etrafındaki dönme hızını; a, robotun hareketli ekseniyle sabit eksen arasındaki açı farkını; L, tekerlek ile robotun ağırlık merkezi arasındaki mesafeyi; n, motorun dişli oranını ve r ise tekerleklerin yarıçapını temsil etmektedir. c_1 ve c_2 sabit değerlerdir.

Şekil 4.6'da örnek olarak birinci tekerin açısal hızı olan $\dot{q}_1$'in FPGA ortamında tasarlanmış hali gösterilmiştir. İkinci ve üçüncü tekerleğin dönme hızları olan $\dot{q}_2$ ve $\dot{q}_3$' de benzer şekilde tasarlanmıştır.



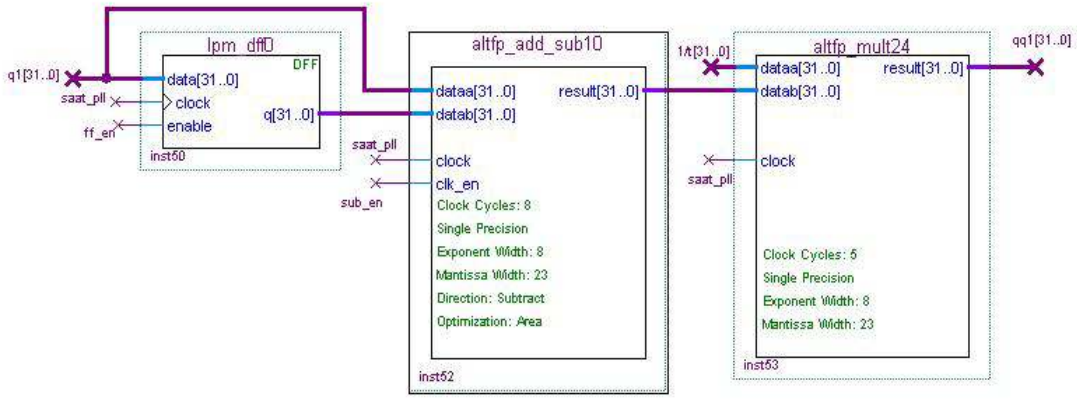
Şekil 4.6. Birinci tekerleğe ait açısal hız

4.4. İvme Hesabı

Dinamik modelde kullanılmak için her bir motorun ivmesinin hesaplanmasına ihtiyaç duyulur. Bunun için hesaplanmış olan açısal hızları kullanılarak her bir tekerleğin ivmesi elde edilmektedir. Örnek olarak aşağıda birinci tekere ait ivme hesabı gösterilmiştir.

$\ddot{q}_1$= ($\dot{q}_1$ (h) - $\dot{q}_1$ (h-1)) /t Burada qq1 birinci tekerleğe ait ivmedir ve $\dot{q}_1$ ise açısal hızıdır, t ise örnekleme süresidir.

Şekil 4.7'de birinci tekerleğin ivme hesabı için gerçekleştirilen FPGA tasarımı gösterilmiştir. İkinci ve üçüncü tekerleğin ivmeleri olan $\ddot{q}_2$ ve $\ddot{q}_3$'de benzer şekilde tasarlanmıştır.



Şekil 4.7. Birinci tekerleğe ait ivme hesabı.

4.5. PI Denetleyici Tasarımı

Gerçekleştirilen tasarımda mobil robotun izlenmesi istenilen yörüngede hareketini tamamlaması için tekerlek hız değerlerinin referans hız değerlerinde olması gerekmektedir. Bu hız değerlerini değiştirecek faktörlere karşı sisteme PI denetleyici eklenmiştir. Robotun qg gerçek hız değerleri q referans değerleri ile PI denetleyiciye girilerek dinamik model ve motor modeli için gerekli hız değeri hesaplanmaktadır. Her bir tekerlek için ayrı ayrı PI denetleyici tasarlanmıştır. Örnek olarak Şekil 4.8'de birinci tekerleğe ait FPGA ortamında PI denetleyici tasarımı gösterilmiştir.

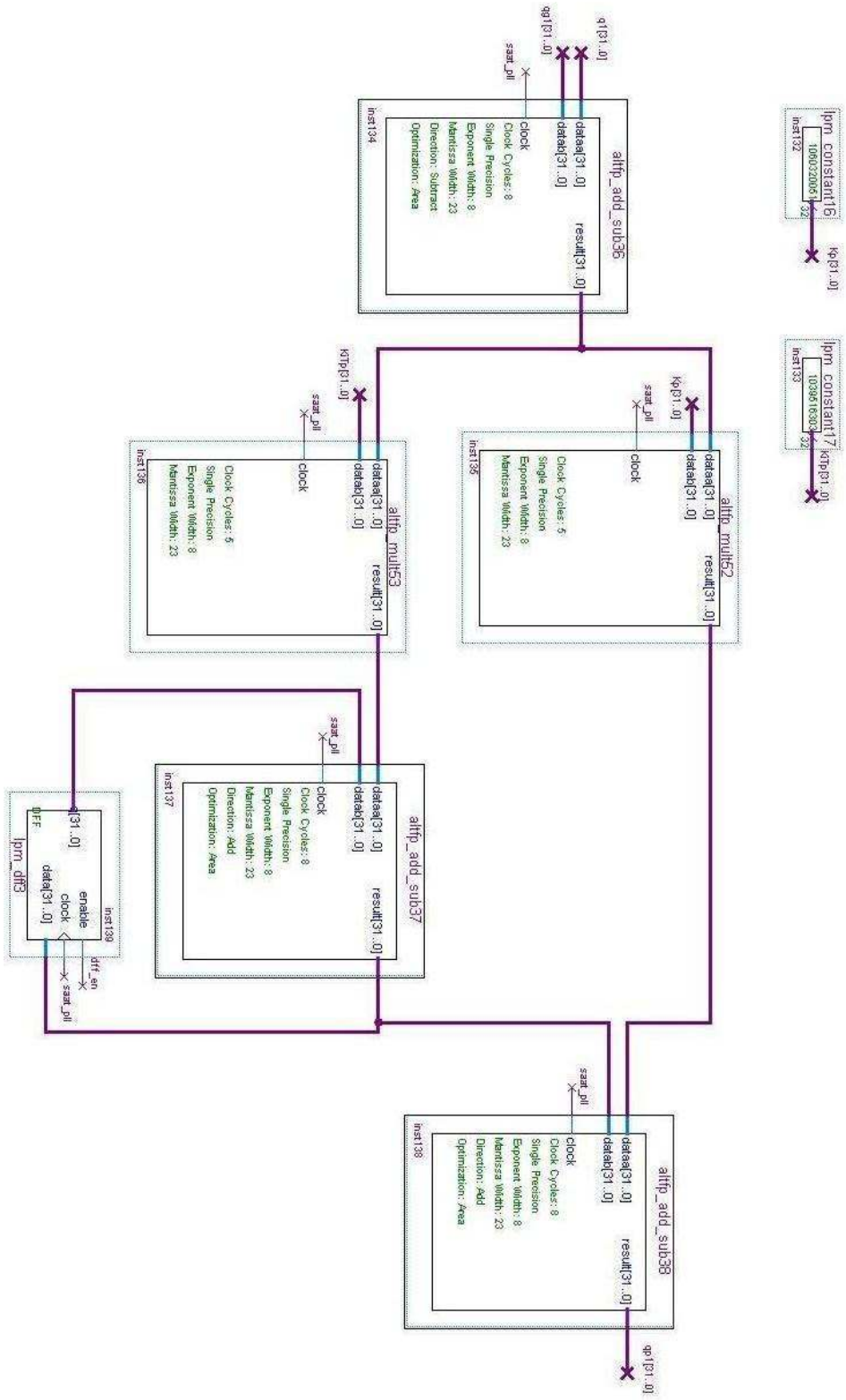
$\dot{q}_1$ birinci tekerleğin referans hızı qg1 ise gerçek hızı olmak üzere; dinamik ve motor modeli için PI çıkışı hız değeri qp1;

qp1=($\dot{q}_1$-qg1)*Kp+($\dot{q}_1$-qg1)*Ki*Tp olur.

Burada Kp oransal katsayı değeri (Kp=0.7), Ki integral katsayı değeri (Ki=60) ve Tp sistemin gerçek zaman değerini (Tp=0.002 sn) göstermektedir. Tasarlanan üç PI denetleyici için de Kp, Ki, ve Tp değerleri aynıdır.

Oluşturulan robot modelinin çevrim süresi 880 clocktur (8.8 µs). Ancak PI denetleyici, robot modelinin her çevrim süresinde 10 çevrim yapmaktadır. Bu nedenle PI denetleyici ve onun önüne bağlı modüllerin çevrim süresi 88 clocktur (0.88 µs).

Şekil 4.8 incelendiğinde PI denetleyiciye gelen q1[31..0] her 8.8 µs'de değişmektedir. Ancak sistemin çıkışından gelen qg1 gerçek hız değeri ise 0.88 µs'de değişmektedir.

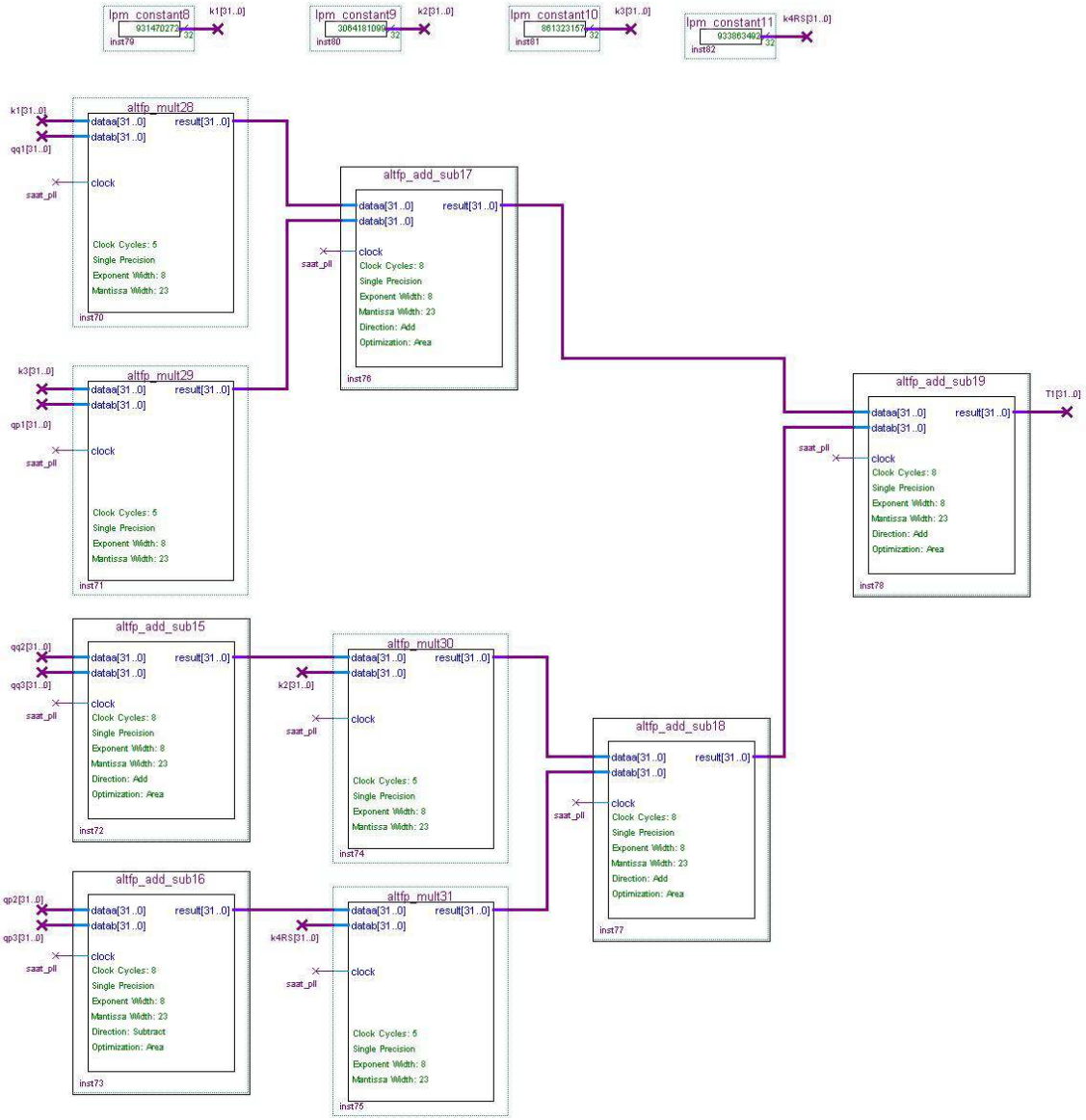


Şekil 4.8. Birinci tekerleğe ait FPGA ortamında PI denetleyici tasarımı

4.6. Moment Hesabı

Robotun her bir tekerleğinin bağlı olduğu motora uygulanması gereken gerilim miktarlarını hesaplamak için ilk olarak her bir motor için moment (tork) hesabı yapılır. Bunun için bölüm 3.3'de anlatılan mobil robot dinamik modelinde gerekli formüller ve açıklamalar ayrıntılı olarak verilmiştir. Dinamik model formülleri doğrultusunda birinci tekerleğe bağlı motorun moment hesabı FPGA tasarımı Şekil 4.9'da gösterilmiştir.

İkinci motorun ve üçüncü motorun momentleri olan T2 ve T3 de benzer şekilde tasarlanmıştır.

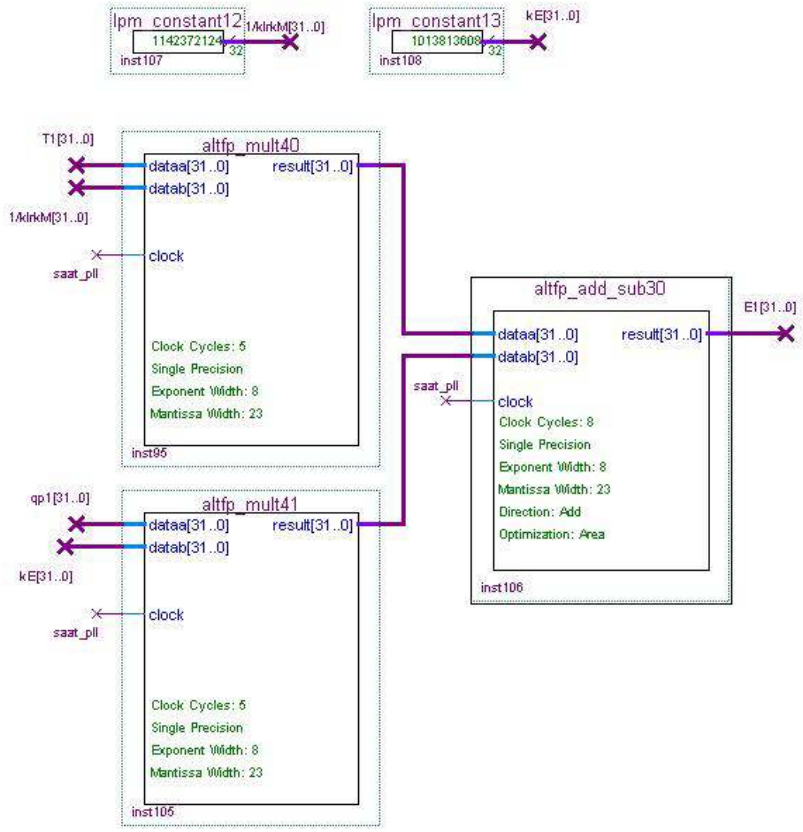


Şekil 4.9. Birinci tekerleğe ait FPGA moment hesabı

4.7. Gerilim Hesabı

Mobil robotun istenilen hızlarda hareket etmesi için gerekli momenti üretecek Tekerlek motorlarına verilmesi uygun gerilim motor modeline göre hesaplanmaktadır. Birinci motorun gerilim değeri olan E1 = 1/(klr*kM)*T1 + kE*$\dot{q}_1$ şeklinde elde edilir. Detaylı motor modeli EK-3'de verilmiştir.

İkinci motorun ve üçüncü motorun gerilimleri olan E2 ve E3 de benzer şekilde tasarlanmıştır. Şekil 4.10'da birinci tekerlek için gerilim hesabının FPGA tasarımı gösterilmiştir.



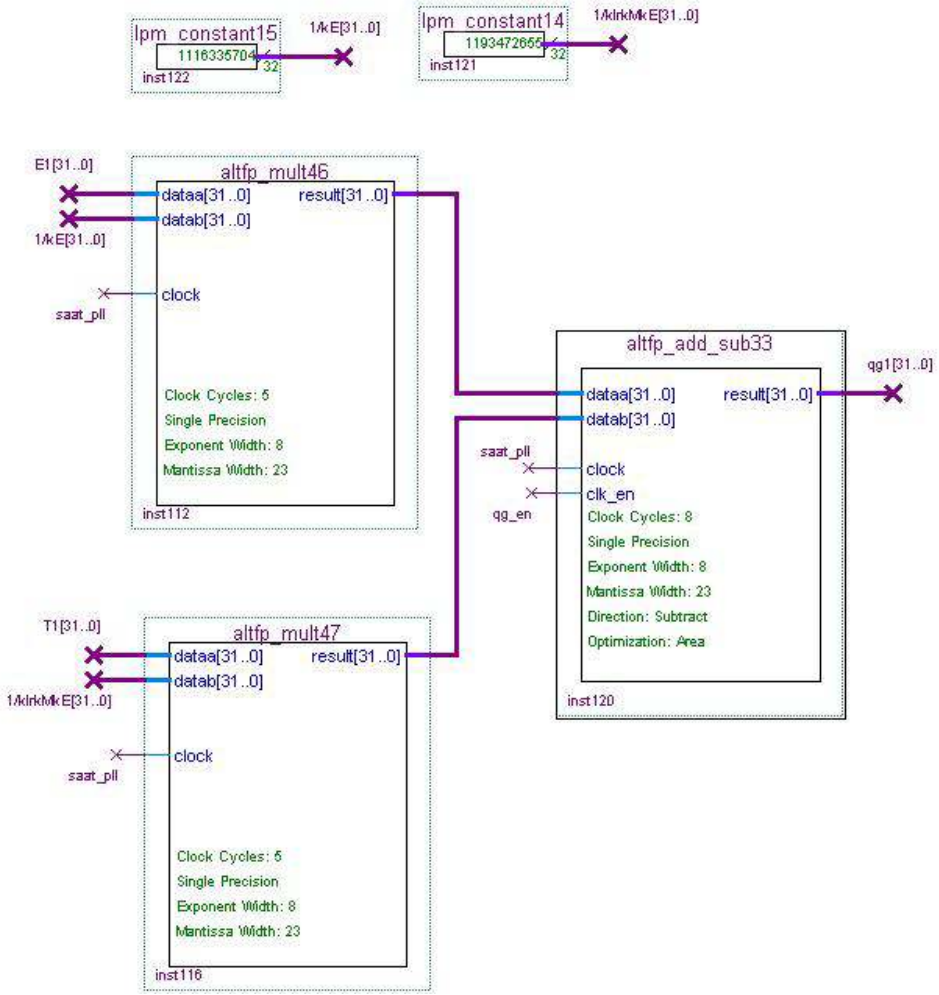
Şekil 4.10. Birinci motora ait gerilim hesabı

4.8. Ters Dinamik Hesabı

Bu bölümde PI denetleyici için mobil robotun her bir tekerleğinin gerçek hız değerleri hesaplanmaktadır. Tekerleklere uygulanan gerilim ve momentten robotun o anki gerçek hız değerlerine ulaşılır. Elde edilen gerçek hız değerleri kapalı döngü oluşturacak şekilde PI denetleyicinin girişine verilmektedir.

Birinci tekerleğe ait gerçek hız değeri qg1=E1/kE-T1/(klr*kM*kE) şeklinde elde edilir.

İkinci tekerleğin ve üçüncü tekerleğin gerçek hız değerleri olan qg2 ve qg3'te benzer şekilde tasarlanmıştır. Şekil 4.11'de birinci tekerleğin gerçek hız değerinin hesaplanmasının Quartus'ta FPGA tasarımı gösterilmiştir.



Şekil 4.11. Birinci tekeleğe ait gerçek hız değerinin hesabı

5. SİMÜLASYON SONUÇLARI

Bu bölümde omni directional mobil robot modelinin farklı yörüngelerdeki robot hareketi için Matlab ve FPGA ortamında gerçekleştirilen simülasyon sonuçları verilmiştir. Robotun belirlenmiş daire ve kare yörüngeleri için gerçekleştireceği hareketin Matlab ve FPGA ortamlarında modellemesi yapılmıştır.

Sistem modeli önce Matlab ortamında gerçekleştirilmiş olup daha sonra aynı sistem FPGA ortamında tasarlanarak sonuçlar değerlendirilmiştir.

Bu tez çalışmasında daha önce Ohio Üniversitesinde benzer yapıda ve benzer özellikteki bir robotun kullanıldığı, Jianhua Wu tarafından hazırlanan doktora tezi çalışmasında deneysel olarak ölçülmüş olan robota ait parametre değerleri kullanılmıştır [7]. Tablo 5.1 ve tablo 5.2'de verilen robot ve motora ait parametre değerleri bu çalışma doğrultusunda deneysel olarak elde edilmiştir.

Tablo 5.1. Mobil robotun parametreleri

J_m (kgm^2)	J_L (kgm^2)	n	c_m (N s/m)	c_L (N s/m)	J (kgm^2)
$2.7e-7$	$8e-5$	14	$5e-8$	0	$4.6e-3$
m (kg)	r (m)	L (m)	k_{lr}	k_M	k_E
2.36	$2e-2$	$7e-2$	1/8.71	0.0144	0.0145

Tablo 5.2'de mobil robot için EOM (Equations of Motion) değerleri verilmiştir.

Tablo 5.2. Robot için EOM katsayıları

k_1	k_2	k_3	k_4
$2.82e-6$	$-8.572e-7$	$5.02e-8$	$7.86e-7$
$k_1/(k_{lr}k_M)$	$k_2/(k_{lr}k_M)$	$k_1/(k_{lr}k_M)+k_E$	$k_4/(k_{lr}k_M)$
$1.6e-3$	$-4.81e-4$	$1.45-e2$	$4.41e-4$

Uygulama 1: **Kare Yörüngesini İzleme**

Bu uygulamada mobil robotun kendi ekseni etrafında dönme hareketi yapmaksızın Şekil 5.1(a)'da gösterilen kare yörüngesini takip etmesi amaçlanmıştır.

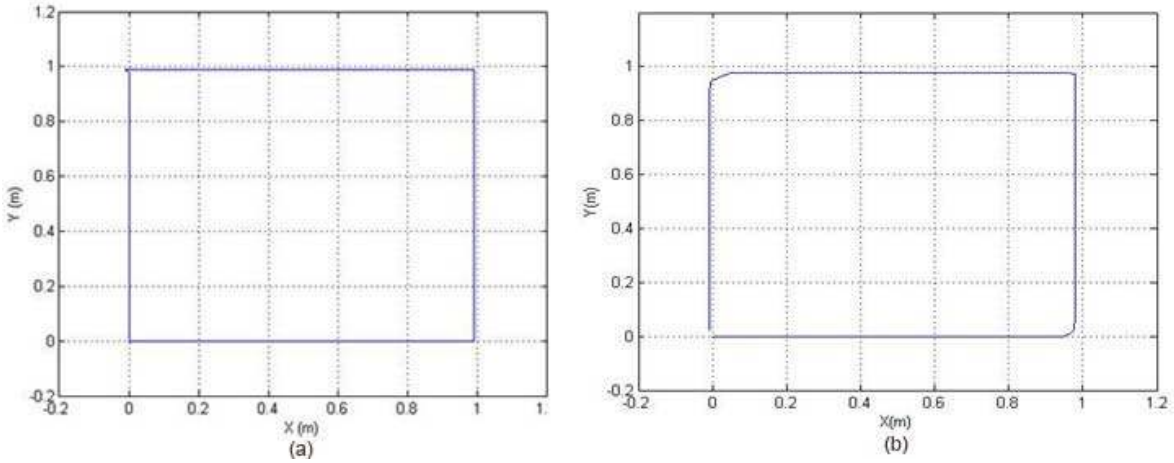
Mobil robotun X ve Y eksenli düzlemde kenar uzunluğu 1m olan kare yörüngesinde kendi üzerinde dönmeksizin hareketi sağlanmıştır. İzlenilmesi istenen karenin toplam çevresi 360 küçük birime ayrılmıştır. Robotun hareketini 7.2 sn'de tamamlaması planlanmış ve buna göre her bir birimde olması gereken hız değerleri hesaplanmıştır.

Robotun her 360 birimdeki hız değeri ve her birimdeki sabit eksen ile hareketli eksen arasındaki açı değişim değerleri FPGA ortamında oluşturulan ROM hafıza alanlarında tutulmuştur. Yapılan bu uygulamada robot kendi ekseni üzerinde dönmediğinden hareketli eksen ile sabit eksen arasındaki açı ve açısal hız değerleri sıfır çıkmaktadır. FPGA ve Matlab ortamında gerçekleştirilen uygulamalarda aynı PI denetleyici katsayıları kullanılmıştır. Bu katsayı değerlerinden; oransal katsayı değeri Kp=0.7, integral katsayı değeri Ki=60 ve sistemin gerçek zaman değeri Tp=0.002 ms'dir. Şekil 5.1(b)'de robotun FPGA tasarımı sonucu izlemiş olduğu yörünge verilmiştir. Mobil robotun izleyeceği kare yörüngesine bağlı olarak Matlab ve FPGA ortamında tasarlanan simulasyon sonuçlarına göre:

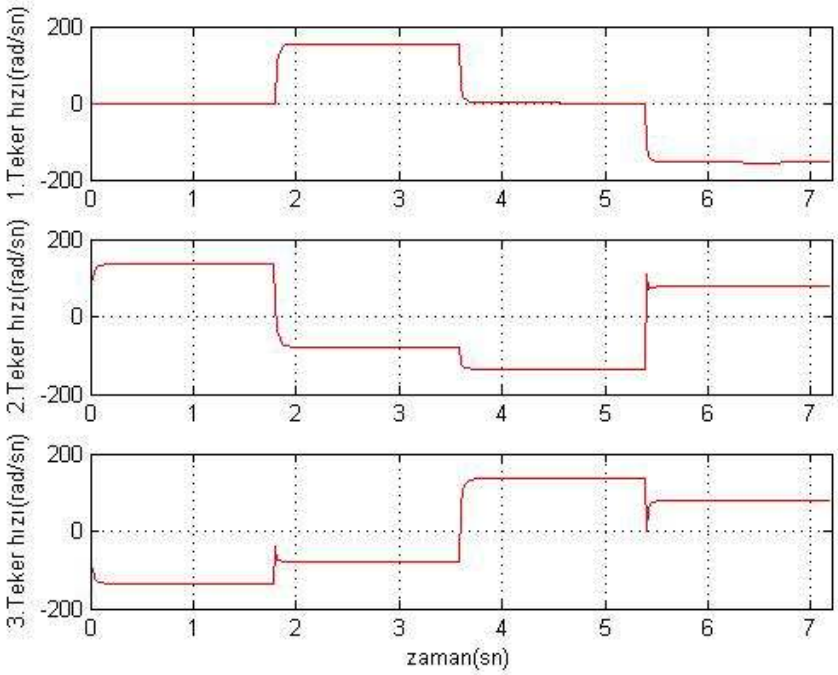
Şekil 5.2 ile Şekil 5.3'de; FPGA ve Matlab simülasyonları sonucunda üç tekere ait açısal hız zaman grafiği verilmiştir.

Şekil 5.4 ile şekil 5.6'da; FPGA ve Matlab simulasyonu sonucunda 1. tekerleğe ait açısal hız zaman grafiği verilmiştir.

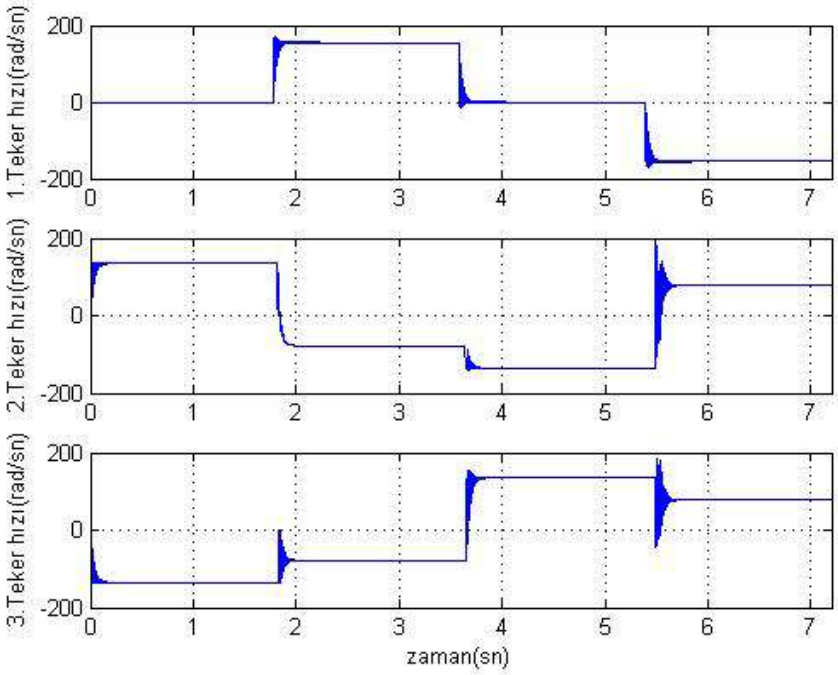
Şekil 5.5 ile şekil 5.7'de; şekil 5.4 ve şekil 5.6'de verilen 1. tekerleğe ait açısal hız zaman grafiğinde kare içine alınan kesitlerin daha detaylı görünmesi için büyütülmüş halleri verilmiştir.



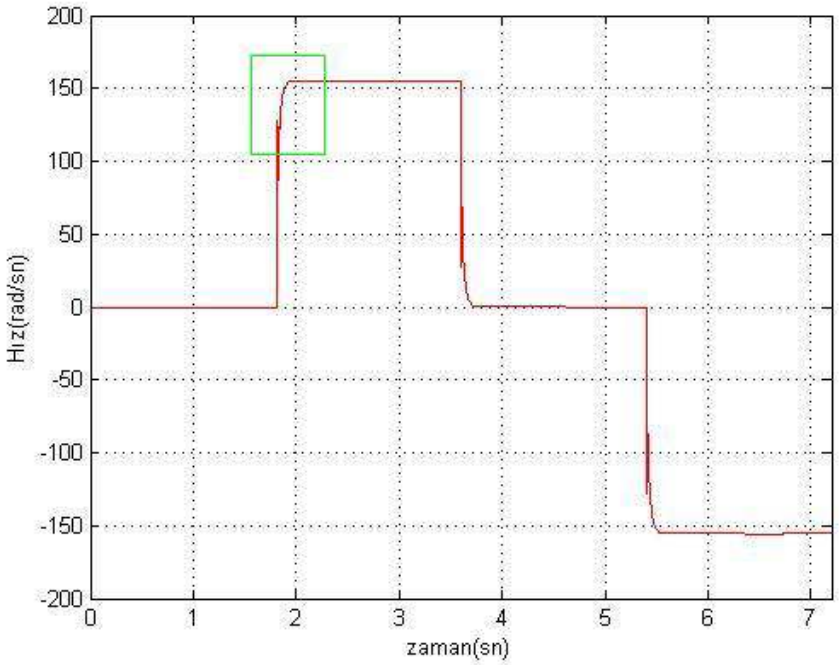
Şekil 5.1. (a) Mobil Robotun izleyeceği kare yörüngesi, (b) FPGA tasarımı sonucu izlemiş olduğu yörünge



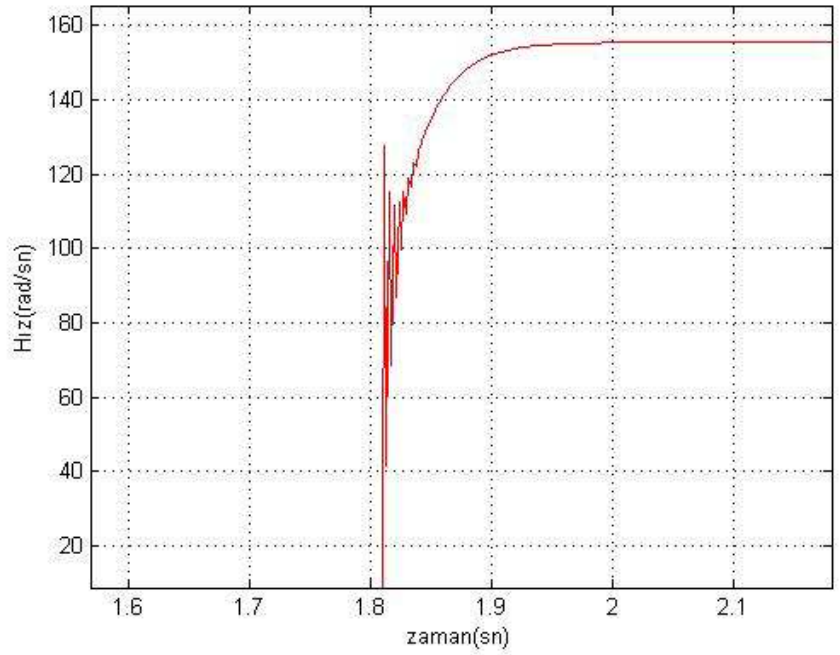
Şekil 5.2. Uygulama 1'in Matlab simülasyonu sonucunda elde edilen tekerleklere ait açısal hızlar.



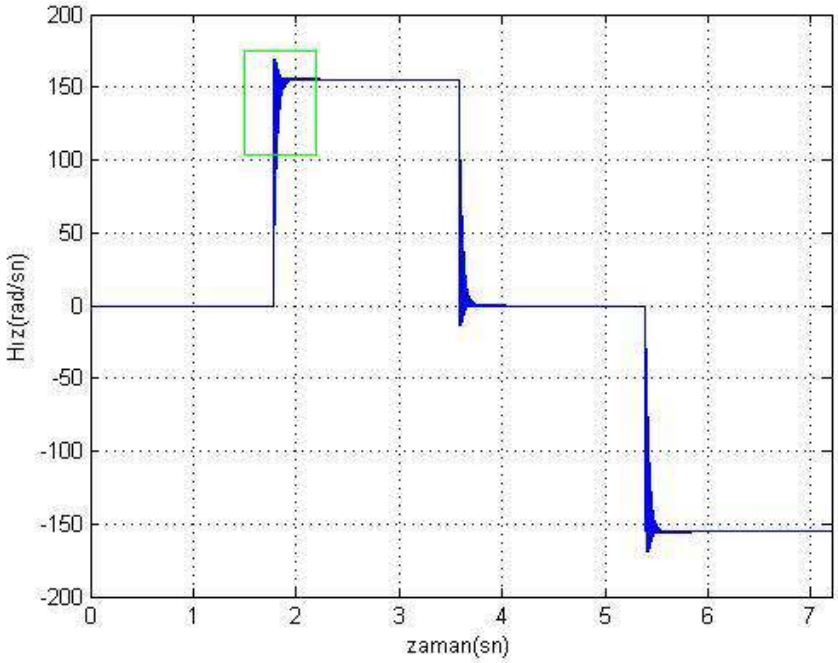
Şekil 5.3. Uygulama 1'in FPGA simülasyonu sonucunda elde edilen tekerleklere ait açısal hızlar



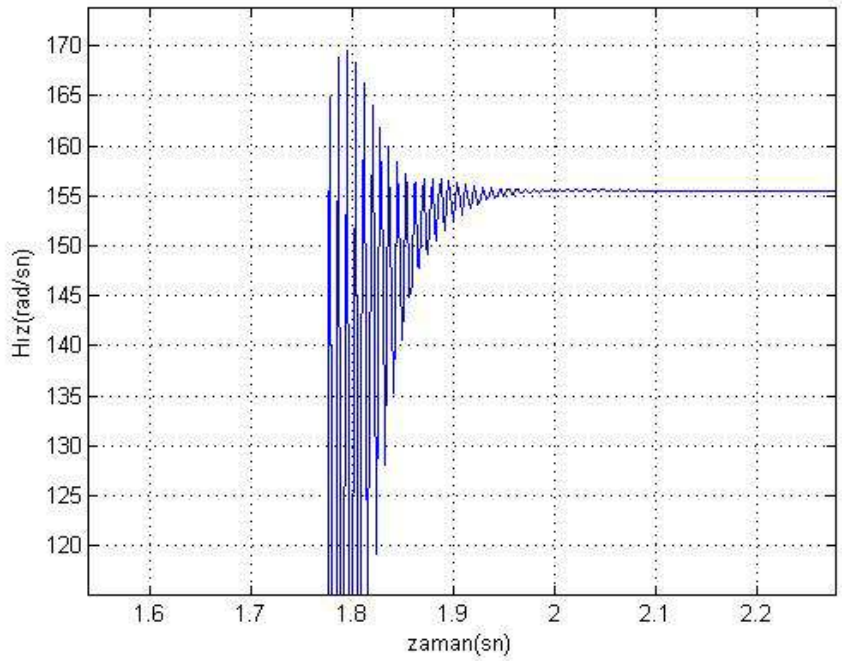
Şekil 5.4. Uygulama 1'in Matlab simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hız



Şekil 5.5. Uygulama 1'in Matlab simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hızın belirli kesitinin (Şekil 5.4'te gösterilen) büyütülmüş hali



Şekil 5.6. Uygulama 1'in FPGA simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hız



Şekil 5.7. Uygulama 1'in FPGA simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hızın belli bir kesitinin (Şekil 5.6'da gösterilen) büyütülmüş hali

Uygulama 2: **Daire Yörüngesini İzleme**

Bu uygulamada mobil robotun kendi ekseni etrafında dönme hareketi yaparak Şekil 5.8(a)'da gösterilen daire yörüngesini takip etmesi amaçlanmıştır.

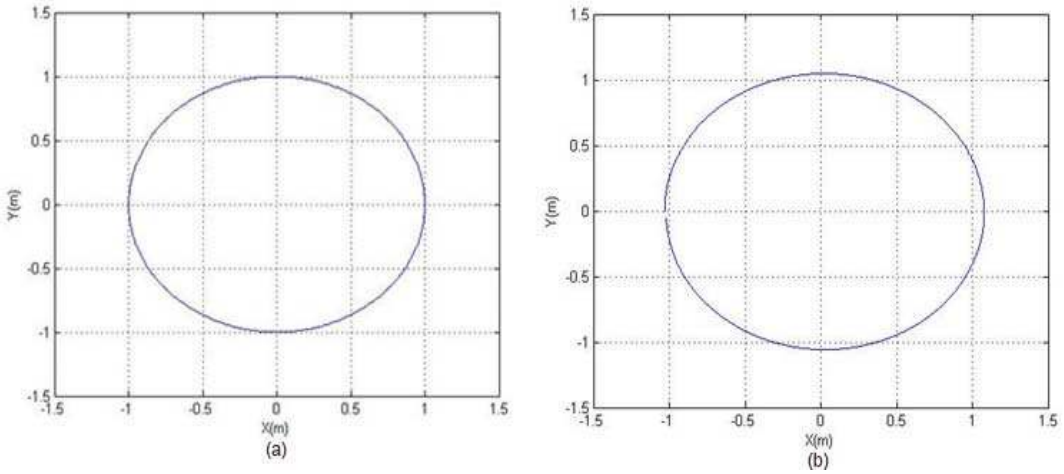
Mobil robotun X ve Y eksenli düzlemde yarıçap uzunluğu 1m olan daire yörüngesinde kendi üzerinde dönerek hareket etmesi sağlanmıştır. İzlenilmesi istenen dairenin toplam çevresi 360 küçük birime ayrılmıştır. Robotun hareketini 7.2 sn'de tamamlanması planlanmış ve buna göre her bir birimde olması gereken hız değerleri hesaplanmıştır.

Robotun her 360 birimdeki hız değeri ve her birimdeki sabit eksen ile hareketli eksen arasındaki açı değişim değerleri FPGA ortamında oluşturulan ROM hafıza alanlarında tutulmuştur. Robotun hareketi boyunca sabit eksen ve hareketli eksen arasındaki değişim her birimde $2^{o'}$dir. Buda robotun hareketini tamamlama süresinde kendi üzerinde de 2 tur (720^{o}) dönmesi demektir. FPGA tasarımı sonucunda robot şekil 5.8(b)'deki yörüngeyi izlemiştir. Mobil robotun izleyeceği daire yörüngesine bağlı olarak Matlab ve FPGA ortamında tasarlanan simülasyon sonuçlarına göre:

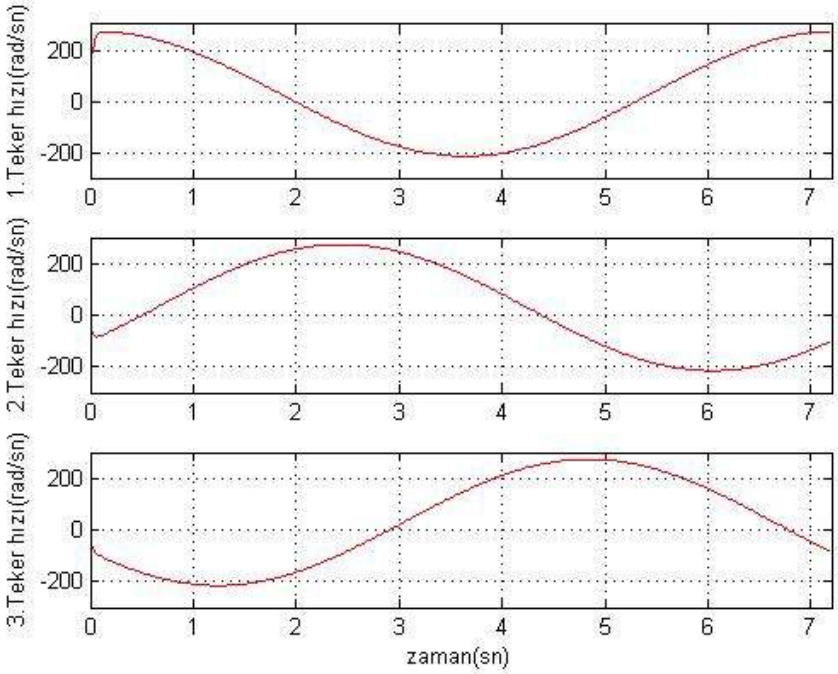
Şekil 5.9 ile Şekil 5.10'da; FPGA ve Matlab simülasyonları sonucunda üç tekere ait açısal hız zaman grafiği verilmiştir.

Şekil 5.11 ile şekil 5.13'de; FPGA ve Matlab simülasyonu sonucunda 1. tekerleğe ait açısal hız zaman grafiği verilmiştir.

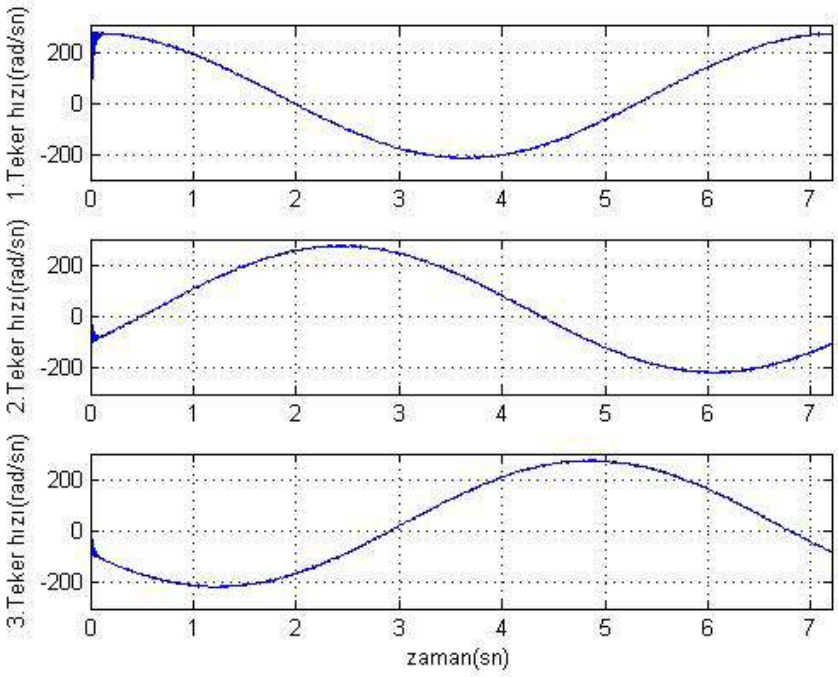
Şekil 5.12 ile şekil 5.14'de; şekil 5.11 ve şekil 5.13'de verilen 1. tekerleğe ait açısal hız zaman grafiğinde kare içine alınan kesitlerin daha detaylı görünmesi için büyütülmüş halleri verilmiştir.



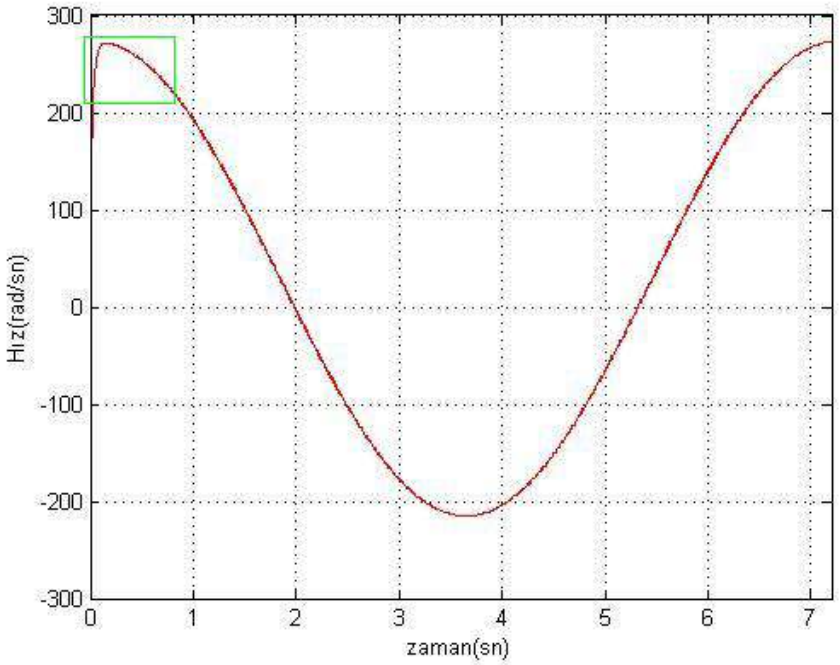
Şekil 5.8. (a) Mobil Robotun izleyeceği daire yörüngesi, (b) FPGA tasarımı sonucu izlemiş olduğu yörünge



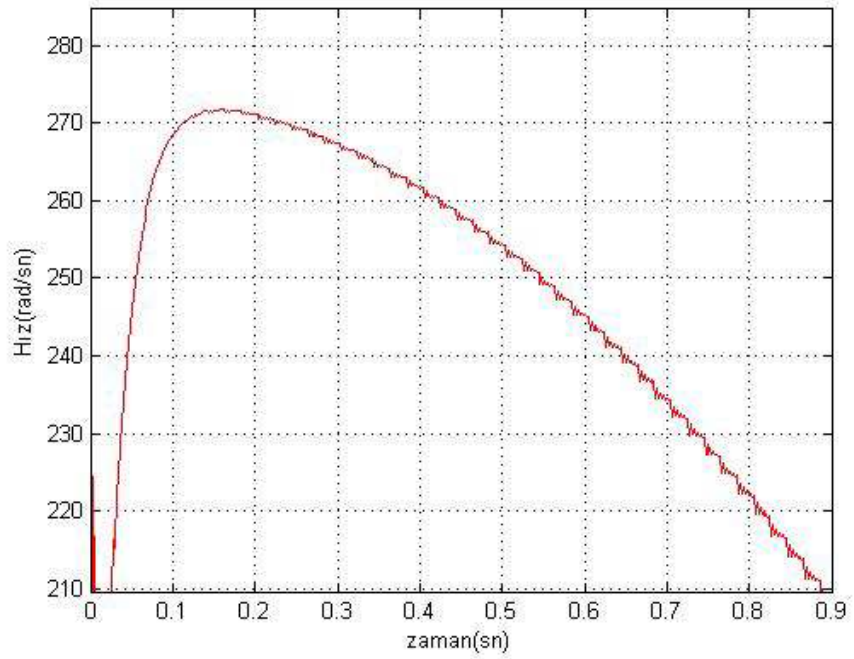
Şekil 5.9. Uygulama 2'in Matlab simülasyonu sonucunda elde edilen tekerleklere ait açısal hızlar



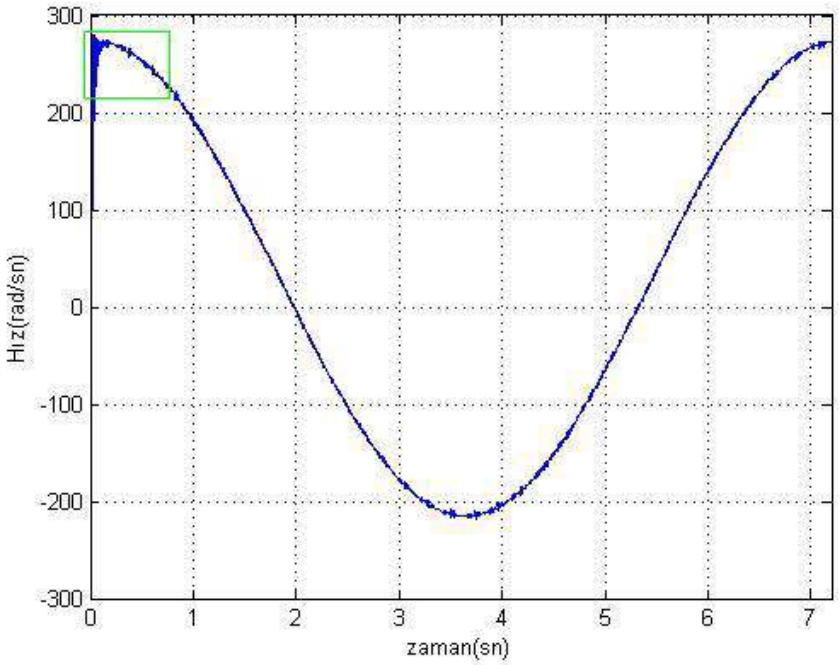
Şekil 5.10. Uygulama 2'in FPGA simülasyonu sonucunda elde edilen tekerleklere ait açısal hızlar



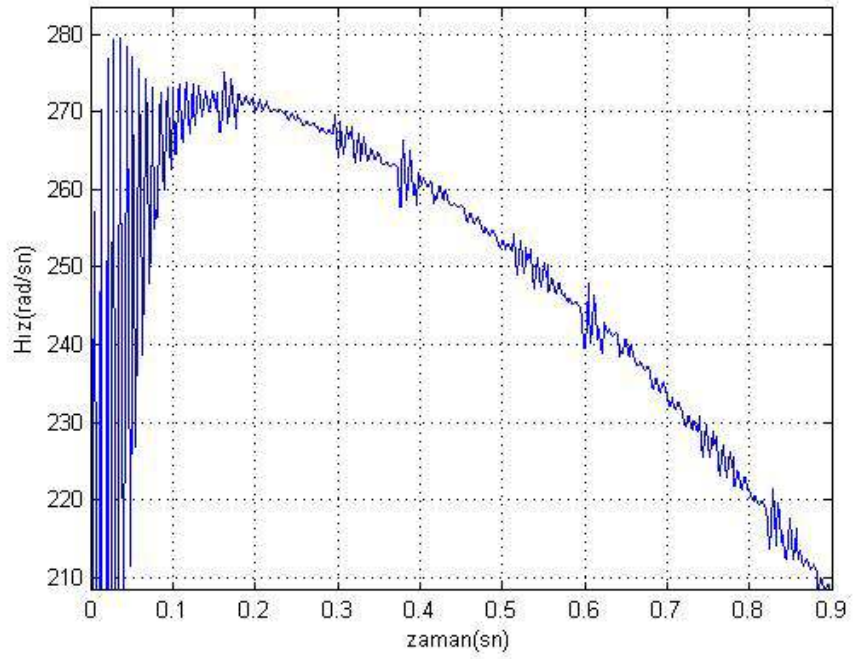
Şekil 5.11. Uygulama 2'in Matlab simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hız



Şekil 5.12. Uygulama 2'in Matlab simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hızın belli bir kesitinin (Şekil 5.11'de gösterilen) büyütülmüş hali



Şekil 5.13. Uygulama 2'in FPGA simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hız



Şekil 5.14. Uygulama 2'in FPGA simülasyonu sonucunda elde edilen 1.tekerleğe ait açısal hızın belli bir kesitinin (Şekil 5.13'de gösterilen) büyütülmüş hali

6. SONUÇLAR

Bu tez çalışmasında gerçek zaman uygulaması olarak bir mobil robot modellemesi ve mobil robotun belirtilen yörüngede hareket etmesi için FPGA tasarımı yapılmıştır. Tekerleklerin hız kontrolü için yine FPGA ortamında tasarlanan PI denetleyici kullanılmıştır.

Omni directional mobil robot tekerleklerinin açısal doğrultuları değiştirilmeden sadece tekerleklerin dönme doğrultuları değiştirilerek robotun istenilen yönde hareketini sağlanabilir. Bu özelliklerinden dolayı günümüzde oldukça geniş kullanım alanına sahiptirler. Tezde referans alınan mobil robot üç bağımsızlık derecesinde hareket edebilen üç tekerlekli bir robottur. Bu yüzden FPGA ortamında her bir tekerlek için ayrı bir PI denetleyici tasarlanmıştır

FPGA Temel lojik işlemlerden karmaşık matematiksel işlemlere kadar pek çok işlevi donanımsal olarak yerine getirir. FPGA yapısında bulunan arabirimler, tasarımdaki bağlantılar göz önüne alınarak elektriksel olarak programlanabilir ve istenildiği kadar programlanabilme yapısına sahip olduğundan tasarımlarda büyük kolaylıklar sağlar. FPGA ile temel mantık kapılarının ve yapısı daha karmaşık olan devre elemanlarının işlevselliği artırılmaktadır. Uzayan tasarım süreçleri, artan maliyet, yenilenme gereksinimleri ve gelişen teknoloji FPGA'ları teknolojide önemli yer sahibi olmasını sağlamıştır. Özellikle yüksek hızda çalışan sistemlerde FPGA'lar oldukça yüksek verim sağlamaktadır. Bunun yanı sıra askeri, uzay, veri işleme, otomotiv gibi teknoloji gerektiren alanlarda da FPGA kullanımı gittikçe yaygınlaşmıştır.

Bu tez çalışmasında üç bağımsız dereceli omni directional mobil robotun tekerlek hızlarının PI denetleyici ile kontrolü Altera firmasının Stratix II GX EP2SGX90FF1508C3 FPGA'sı kullanılarak gerçekleştirilmiştir. FPGA konfigrasyonu Quartus II 8.1 versiyonunda oluşturulmuştur. Oluşturulan modelin bir çevrim süresi 880 clocktur, sistemin çalışma frekansı 10 ns olduğundan sistem bir çevrimi 8.8 µs zaman diliminde gerçekleştirmektedir. Bu zaman dilimi, modelin gerçek zamanda çalıştırılması için oldukça düşük bir değerdir. Buda FPGA'nın yüksek hızlı çalışan sistemlerdeki etkinliğini göstermektedir.

Robot modelinin FPGA simülasyon sonuçları kendi simülatöründe ve Matlab ortamında çizdirilerek incelenmiştir. Böylece Matlab programı ile FPGA simülasyon sonuçları karşılaştırılarak sonuçların doğruluğu kanıtlanmıştır.

FPGA ortamında tasarlanan simülasyon sonuçları ile Matlab ortamında tasarlanan simülasyon sonuçlarının birebir örtüşmemesinin sebeplerinden biri FPGA ortamında kullanılan sayı sistemi ile Matlab ortamında kullanılan sayı sisteminin birebir örtüşmemesindendir. Bir başka sebep; özellikle geçici rejimdeki hız değerlerinin sürekli değişmesinden ve sabit kullanılmamasından dolayı ivme değerlerinin çok büyük değerler alabilmesi olarak açıklanabilir.

Gerçeklenen tasarım sonucunda Stratix II GX EP2SGX90FF1508C3 FPGA'sının toplam kaynaklarından; bileşimsel ALUT:35.963, lojik kayıtçı (logic register):28.076, giriş ve çıkış pini:385, toplam hafıza alanı:128.449 bit ve pll:1 tane tüketilmiştir. Toplam kaynak sayısının kullanılan kaynak oranı ile karşılaştırıldığında; bileşimsel ALUT: 35.963 / 72.768 (%49), lojik kayıtçı (logic register): 28.076 / 72.768 (%39), toplam giriş ve çıkış pinleri: 385 / 739 (%52)'dir.

Tez çalışması sonunda yüksek hızlı işlem gerektiren gerçek zaman uygulamalarında FPGA kullanımının sistemin daha verimli çalıştıracağı gözlemine varılmıştır.

KAYNAKLAR

[1] **Karris, S.T.,** 2005. Digital Circuit Design with an Introduction to CPLDs and FPGAs, United States of America.

[2] **Mayer, U.,** 2008. Digital Signal Processing with Field Programmable Gate Arrays, USA..

[3] **Hamblen, J.O., Hall, T.S. and Furman M.D.**, 2005. Rapid Prototyping of Digital Systems, Kluwer Academic Publishers, United States of America.

[4] **Siegwart, R., Nourbakhsh, I.R.,** 2004. Introduction to Autonomous Mobile Robots, London.

[5] **Braunl, T.,** 2008. Mobile robot Design and Aplication with Embedded System, Australia.

[6] **Wu, J.,** 2005. Dynamic Path Planning of an Omni-Directional Robot in a Dynamic Environment, PhD. Thesis, The Fritz J. And Dolores H. Russ College of Engineering and Technology of Ohio University.

[7] **Henning, T.P.,** 2003. Dynamics And Controls For A Omni-Direcitonal Robot, Master Thesis, The Fritz J. And Dolores H. Russ College of Engineering and Technology of Ohio University.

[8] **Haendel, R.P.A.,** 2005. Design of an omnidirectional universal mobile platform, Master Thesis, Eindhoven University of Technology Department of Mechanical Engineering, Eindhoven.

[9] **Özdemir, G.,** 2008. Mobil Robotlarda Programlanabilir Kapı Dizileri Alanı Kullanılarak Gerçek Zamanlı Modelleme, Yüksek Lisans Tezi, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ.

[10] **Aydın, A.,** 2005. FPGA Yonga Mimarisi ve Kullanımı, Yüksek Lisans Tezi, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü, Isparta.

[11] **Kazancı, F.,** 2002. FPGA'lerle Mikroişlemci Tasarımı ve Gerçeklenmesi, Yüksek LisansTezi, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[12] **Şahin, S.,** 2005. FPGA ile Yapay Sinir Ağının Donanımsal Gerçeklenmesi, Yüksek Lisans Tezi, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Fen Bilimleri, Kocaeli.

[13] Howe, T.K., 2003. Evaluation of the Transient Response of a DC motor using Matlab Simulink, Master Thesis, School of Information Technology and Electrical Engineering University of Queensland, Australia.

. **[14] Lin, K.H., Lee, H.S., Chen, W.T.,** 2008. Implementation of Obstacle Avoidance and ZigBee Control Functions for Omni Directional Mobile Robot, IEEE International Conference on Advanced Robotics and its Social Impacts, Taipei, Taiwan.

[15] Lee, J.H., Jung, S., 2008, Global Position Tracking Control of an Omni-directional Mobile Robot Using Fusion of a Magnetic Compass and Encoders, , IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Seoul, Korea.

[16] Tu, K.Y., Shi, C.L., 2006. Design and Implementation of Omni-Directional Soccer Robots for RoboCup, IEEE International Conference on Systems, Taipei, Taiwan.

[17] Soygüder, S., Alli, H., 2006. Çok Yönlü Tekerleklere Sahip Bir Mobil Robotun PLC ile Denetimi, Fırat Üniv. Fen ve Müh. Bil. Dergisi, Elazığ, 403-412.

[18] Samani, H.A., Abdollahi A., Ostadi, H., Rad, S.Z., 2004. Design and Development of a Comprehensive Omni directional Soccer Player Robot, International Journal of Advanced Robotic Systems, ISSN 1729-8806.

[19] Watanabe, K., 1998. Control of an omnidirectional Mobile Robot, Second International Conferem on Knowledge-Based Intelligent Electronic Systems, Adelaide, Australia

[20] İzumi, K., Watanabe, K., Miyazaki T., 1998. Fuzzy Behavior- Based Control for a Miniature Mobile Robot, International Conference on Knowledge-Based Intelligent Electronic System, Adelaide, Australia.

[21] Watanabe, K., Shiraishi, Y., Tzefestas, S.G., Tang, J., Fukuda, T., 1998. Feedback Control of an Omnidirectional Autonomous Platform for Mobile Sevice Robots, Journal of İntelligent and Robotic Systems 22:315-330.

[22] Tang, J., Watanabe, K., Shiraishi, Y., 1998. Design and Traveling Experiment of an Omnidirectional Holonomic Mobile Robot, IEEE, 0-7803-3213.

[23] Ribeiro, F., Moutinho, I., Silva, P., Fraga, C., Pereira, N., 2004. Controlling Omni-directional Wheels of a MSL RoboCup Autonomous Mobile Robot,Universidade do Minho, Portugal.

[24] **Carter, B., Good, M., Lew, J., Williams, R.L., Gallina, P.,** Mechanical Design and Modeling of an Omni-directional RoboCup Player, Department of Mechanical Engineering Ohio University, USA

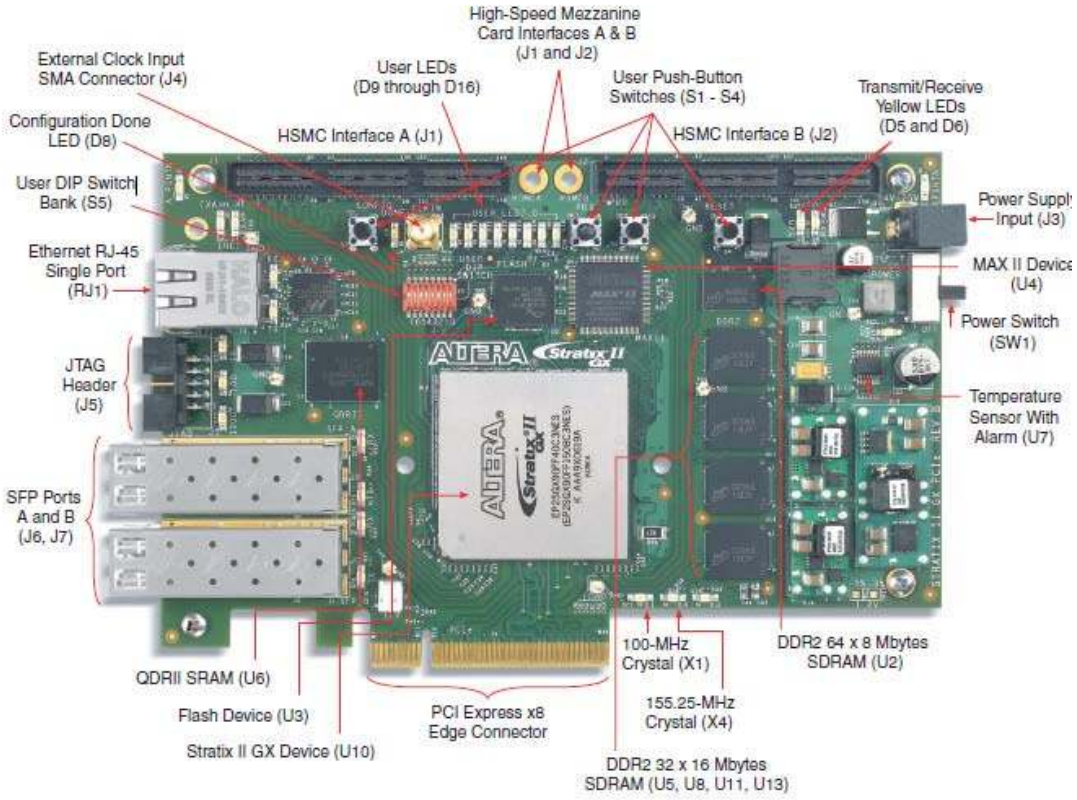
[25] **Wolf, D.F., Holanda, J.A., Bonato, V., Peraon, R., Marques, E.,** 2007. An FPGA Based Mobile Robot Controller, IEEE, 1-4244-0606-4.

[26] **Dick, C.,** 2000. The High and Alternative for DSP Aplications, DSP Engineering, Spring.

[27] **www.altera.com,** Altera Corporation, 1 Kasım 2009.

EK-1. FPGA GELİŞTİRME KARTI VE ÖZELLİKLERİ

Bu tez çalışmasında, Altera firmasının üretmiş olduğu Stratix ailesinden Stratix II GX model FPGA kullanılmıştır. Bu bölümde kullandığımız FPGA ve geliştirme kartının teknik özellikleri verilecektir. Şekil EK-1.1 'de kullanılan geliştirme kartının yukarıdan görünüşü verilmiştir.



Şekil EK-1.1 FPGA Geliştirme Kartı

Geliştirme kartının teknik özellikleri şu şekilde sıralanabilir.

- Yonga dışı hafıza
  - DDR2 SDRAM (256 Mbyte - 72 bit genişliğinde 32 Kbyte derinliğinde)
  - QDRII SRAM (18 Mbyte – 36 bit genişliğinde 512 Kbyte derinliğinde)
- FPGA konfigürasyonu
  - MAX® II CPLD ve 16 bit sayfa modlu kalıcı bellek (512 Mbyte)
  - JTAG arayüzü
- Kullanıcı ve karta özgü arayüzler

- Push-button anahtarları
- Kullanıcı DIP switch
- Kullanıcı LEDleri
- Borda özgü DIP switch (FPGA konfigürasyon ayarlarını kontrol eder.)
- Borda özgü LEDler

➢ Güç kaynağı
- Bileşenlerin gücü
- Rayın gücü
- Ana güç kaynağı
  - PCI anakart

➢ Haberleşme portları
- PCI kenar konnektörü (8 kanal)
- Yüksek hızlı ara kat kartları
- Gigabit ethernet
- SFP modülleri
- JTAG bağlantısı (JTAG tabanlı FPGA haberleşmesi için 10 pin bağlantı)

➢ Clock Devresi
- Üç yüksek hızlı clock osilatörü
  - 100 MHz
  - 155.52 MHz
  - 156.25 MHz
- Harici clock giriş ve çıkışı için SMA konektör

FPGA'ın teknik özellikleri şu şekilde sıralanabilir.

➢ 1.2-V VCCINT
➢ 1.2-V - 3.3-V VCCIO
➢ 1.2-V - 1.5-V I/O alıcı/verici gücü
➢ 78 eşzamanlı kaynak kanalları
➢ 132,540 lojik elementi (LE)
➢ 8 PLL
➢ 798 kullanıcı giriş çıkışı
➢ 6,747,840 RAM biti
➢ 252 18x18 çarpma ünitesi

EK-2. KAYAN NOKTALI (FLOATİNG POİNT) SAYILARDA DÖRT İŞLEM

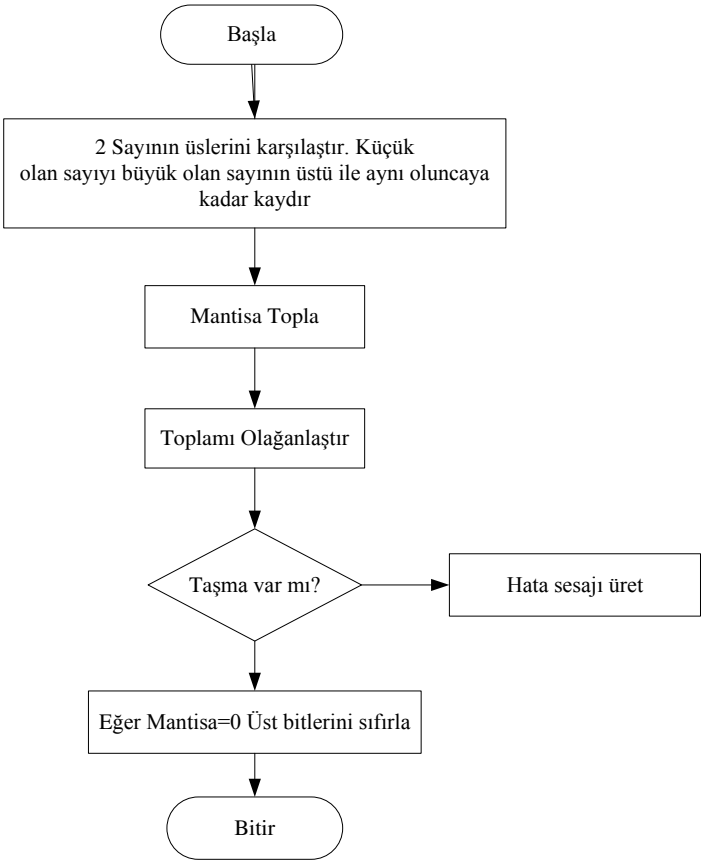
Bu bölümde kayan noktalı (floating point) sayılarla yapılan dört işlem algoritmaları verilmiştir.

EK-2.1. Kayan Noktalı Gösterimde Toplama

Kayan noktalı sayılarda toplama yapmak sabit noktalı sayılarda toplama yapmaktan çok farklıdır, toplama tek aşamada değil birden çok aşamada gerçekleştirilir:

1. Üstler karşılaştırılarak aralarındaki fark bulunur.
2. Eğer üstler farklı ise küçük üsse ait olan mantis, fark kadar sağa kaydırılır. Böylece toplanacak iki sayının üsleri eşitlenmiş olur.
3. Kaydırılmış mantisler toplanır, Sonuç işareti belirlenir.
4. Sonuç mantisi normalize (olağanlaştırma) edilerek 1.*f* standardına getirilir. Eğer gerekiyorsa üs de arttırılır veya azaltılır.

Şekil EK-2.1'de kayan noktalı sayılarda topla işlemin şematik algoritması verilmiştir.



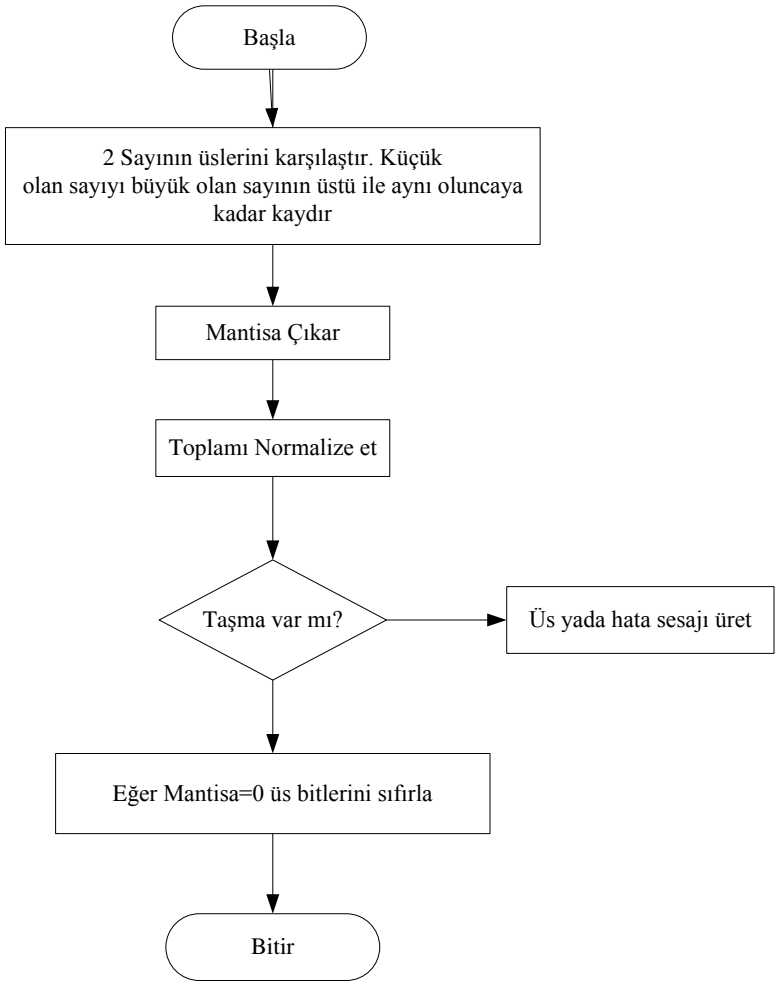
Şekil EK-2.1 Toplama işlemi

EK-2.2. Kayan Noktalı Gösterimde Çıkarma

Kayan noktalı sayılarda çıkarma aşağıda belirtilen aşamalarda gerçekleştirilir:

1. Üsler karşılaştırılarak aralarındaki fark bulunur.
2. Eğer üsler farklı ise küçük üsse ait olan mantis, fark kadar sağa kaydırılır. Böylece toplanacak iki sayının üstleri eşitlenmiş olur.
3. Kaydırılmış mantisler çıkarılır, Sonuç işareti belirlenir.
4. Sonuç mantisi normalize (olağanlaştırma) edilerek 1.*f* standardına getirilir. Eğer gerekiyorsa üs de arttırılır veya azaltılır.

Şekil EK-2.2'de kayan noktalı sayılarda çıkarma işlemin şematik algoritması verilmiştir.



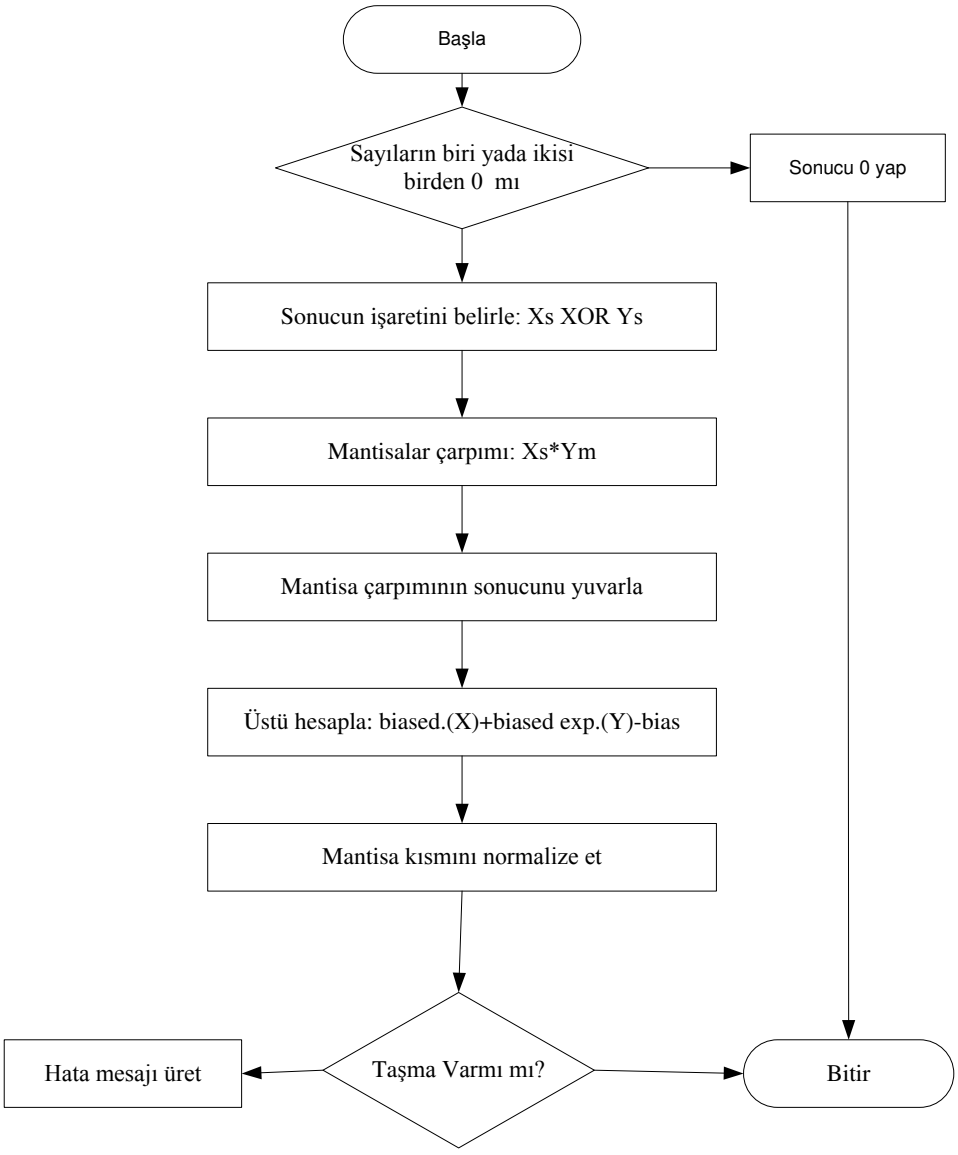
Şekil EK-2.2. Çıkarma İşlemi

EK-2.3. Kayan Noktalı Gösterimde Çarpma

Çarpma işlemleri şu şekildedir:

1. Sonuç işaret biti belirlenir (Xs XOR Ys).
2. Mantisler çarpılır ve sonuç yuvarlanır.
3. Üst hesaplanır
4. Mantis çarpımı normalize (olağanlaştırma) edilerek sonuç mantisi 1.f standardına getirilir.

Şekil EK-2.3'de kayan noktalı sayılarda çarpma işlemin şematik algoritması verilmiştir.



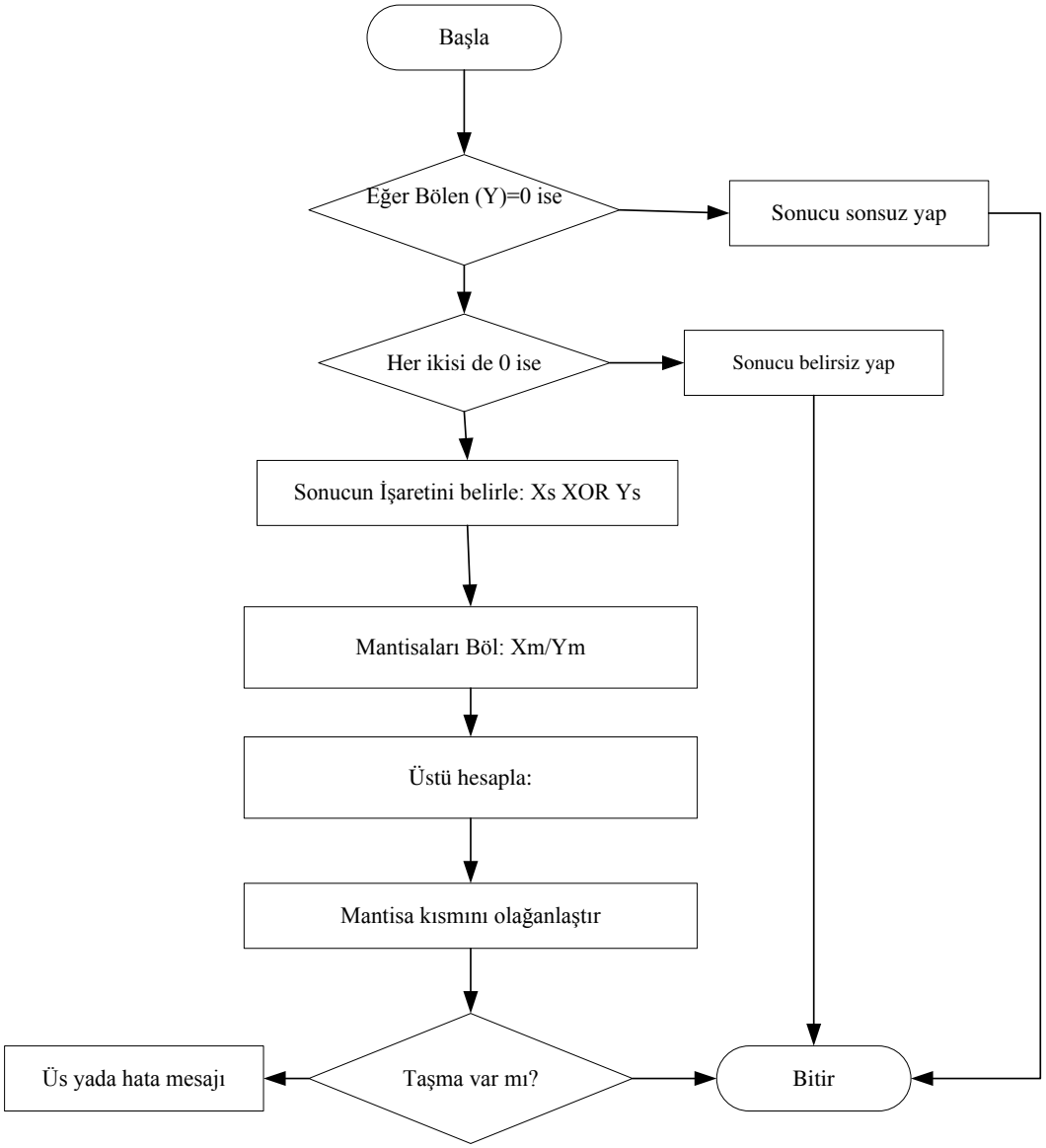
Şekil EK-2.3. Çarpma işlemi

EK-2.4. Kayan Noktalı Gösterimde Bölme

Bölme işlemleri şu şekildedir:

1. Üstler toplanır ve toplamdan yanlılık çıkarılır ve sonuç işaret biti elde edilir.
2. Mantisler bölünür ve sonuç yuvarlanır
3. Üst hesaplanır
4. Mantis kısmını normalize (olağanlaştırma) edilerek sonuç mantisi 1.*f* standardına getirilir.

Şekil EK-2.4'de kayan noktalı sayılarda bölme işlemin şematik algoritması verilmiştir.



Şekil EK2-4. Bölme işlemi

EK-3. DC MOTORLARIN ÇALIŞMA PRENSİBİ VE MATEMATİKSEL MODELİ

DC Motorlar robotik uygulamalarında ve endüstriyel alanda sıklıkla kullanılmaktadırlar. Ucuz ve küçüktürler. Genel olarak 1.5V ile 100V arasında çalışabilirler. 6V, 12V ve 24V motorlar çok yaygın olarak bulunmaktadır. Birkaç bin RPM den on binlerce RPM e kadar çalıştırılabilirler. 12V ve daha küçük motorlar yapısına göre birkaç yüz mili amperden birkaç mili ampere kadar akım çekebilirler. DC motorların genel özellikleri:

- Yüksek hız
- Düsük moment
- Ters yönde kullanım
- Sürekli hareket

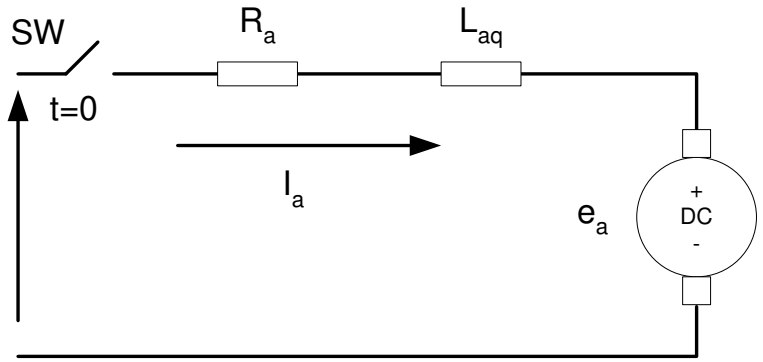
olarak sıralanabilir.

Mantık olarak bobin üzerinden geçen akımın sonucunda oluşturduğu manyetik alanlar sayesinde meydana gelen kutuplaşmayı ileri ve geri yönlü olarak kullanarak yani zıt kutupların çekmesi ya da aynı kutupların birbirini itmesi prensibinin dairesel harekete dönüştürülmesini baz alan en basit yapıdır.

Endüvi dönerken üzerindeki iletkenler de manyetik alan içerisinde döndüklerinden bu iletkenlerde bir endüksiyon elektromotor kuvveti indüklenir. Doğru akım makinesi kullanım amacına göre dinamo ya da motor olarak çalıştırılabilir. Bu formlardan birisinde çalışma, makinede herhangi bir değişikliği gerektirmez. Eğer makine dinamo olarak çalıştırılırsa moment yön değiştirir. Manyetik alan içinde etkin uzunluğu "L" ve içerisinden geçen akımı "i" olan bir iletken akı yoğunluğu B olan bir alan içerisinde kalırsa, iletken manyetik alan tarafından itilir. İletkenin alana dik olma durumunda meydana gelen itme kuvvetinin büyüklüğü "Newton" olarak F=B.i.L olur. Alan tarafından iletken üzerinde oluşturulan itme kuvvetinin yönü iletkenin taşıdığı akımın yönüne bağlıdır. İletkende itme kuvveti olduğu sürece iletkende bir hareket veya dönme olayı meydana gelir .

DC motorlarda endüvi ve uyarma olmak üzere iki farklı sargı vardır. DC motorlar temel olarak uyarma sargısının yapısına göre sabit uyarmalı ve değişken uyarmalı olarak iki farklı şekilde sınıflandırılabilir. Kullanılmakta olan motor sabit uyarmalı kalıcı manyetik motordur. Bu motorlarda uyarma sargısındaki manyetik akı (Φ) sabit tutulur, endüvi geriliminin (V_t) değiştirilmesiyle endüviden akan akım (I_a) değişir. Böylece akımla,

motorda üretilen moment arasındaki ilişkiden moment kontrol edilir. Şekil EK-3.1 'de DC motorun eşdeğer devresi gösterilmektedir.



Şekil EK-3.1 DC Motorun Eşdeğer Devresi

Şekil EK-3.1 'de temsil edilen DC motora ait değişkenler şunlardır:

V_t = Endüvi gerilimi

Laq = Endüvi endüktansı

Ra = Endüvi direnci

e_a = Zıt elektro motor kuvveti

I_a = Endüvi akımı

ω_m = Motorun açısal hızı (raydan/sn)

T = Moment

T_L = Yük momenti

Motor tarafından üretilen zıt elektro motor kuvveti, akıyla ve açısal hızla doğru orantılıdır. Kalıcı manyetik motorlarda sabit uyarma kullanıldığında sabit akı elde edilir. Bu durumda zıt elektro motor kuvvetiyle açısal hız doğru orantılı olarak değişecektir. Bu iki değişken arasındaki orantıyı veren büyüklüğe de zıt elektro motor kuvvet sabiti denir. Aynı şekilde motorda üretilen moment uyarma sargısındaki akı ve endüvi sargılarından akan akım doğru orantılıdır. Sabit akı durumunda aşağıda gösterildiği gibi momentle akım arasında doğrusal bir ilişki ifade edilir.

Şekil EK-3.1'deki DC motorun eşdeğer devresi baz alınarak motorun matematiksel modeli şu şekilde elde edilir.

Manyetik doğrusallık varsayılarak temel motor eşitlikleri şöyle verilebilir:

$$T = K_f i_f i_a = K_m i_a \quad \text{(EK-3.1)}$$

$$e_a = K_f i_f \omega_m = K_m \omega_m \quad \text{(EK-3.2)}$$

Burada $K_m = K_f i_f$, bir sabittir ve aynı zamanda e_a / e_m oranına eşittir.

Eşitlik EK-3.1 ve EK-3.2 'nin Laplace dönüşümleri sonucunda şu eşitlikler elde edilir:

$$T(s) = K_m i_a(s) \quad \text{(EK-3.3)}$$

$$E_a = K_m \omega_m(s) \quad \text{(EK-3.4)}$$

t=0 anında SW switch'i kapatılırsa;

$$V_t = e_a + R_a i_a + L_{aq} \frac{di_a}{dt} \quad \text{(EK-3.5)}$$

Eşitlik EK-2.2, Eşitlik EK-2.5 'de yerine yazılırsa;

$$V_t = K_m \omega_m + R_a i_a + L_{aq} \frac{di_a}{dt} \quad \text{(EK-3.6)}$$

Sıfır başlangıç şartı için Eşitlik EK-2.6 'nın Laplace dönüşümü sonucunda;

$$V_t(s) = K_m \omega_m(s) + R_a I_a(s) + L_{aq} s I_a(s) \quad \text{(EK-3.7)}$$

veya

$$V_t(s) = K_m \omega_m(s) + I_a(s) R_a (1 + s\tau_a) \quad \text{(EK-3.8)}$$

elde edilir. Burada $\tau_a = L_{aq} / R_a$ armatürün elektriksel zaman sabitidir.

Mekanik sistem için dinamik eşitlik;

$$T = K_m i_a = J \frac{d\omega_m}{dt} + B\omega_m + T_L \quad \text{(EK-3.9)}$$

şeklinde verilir. Burada $B\omega_m$ terimi sistemin dönel moment kaybını ifade eder.

Eşitlik EK-3.1 ve Eşitlik EK-3.9 'un Laplace dönüşümleri sonucu;

$$T(s) = K_m i_a(s) = J s\omega_m(s) + B\omega_m(s) + T_L(s) \quad \text{(EK-3.10)}$$

olur.

Eşitlik EK-3.10 ve EK-3.3 'den;

$$\omega_m(s) = \frac{T(s) - T_L(s)}{B(1 + sJ/B)} = \frac{K_m I_a(s) - T_L(s)}{B(1 + s\tau_m)} \quad \text{(EK-3.11)}$$

elde edilir.Burada $\tau_m = J/B$, sistemin mekaniksel zaman sabitidir.

Eşitlik EK-3.4 ve EK-3.8 'den

$$I_a(s) = \frac{V_t(s) - E_a(s)}{R_a(1 + s\tau_a)} = \frac{V_t(s) - K_m\omega_m(s)}{R_a(1 + s\tau_a)} \qquad \text{(EK-3.12)}$$

elde edilir.

EK-4. ROBOT DİNAMİK MODEL PARAMETERE HESABI

Mobil robotun konum vektörü $s_w = [x_m \ y_m]^T$ olarak tanımlanır. Buradan;

$$M\ddot{S}_m = F_w \tag{EK-4.1}$$

Burada $F_w = [F_x \ F_y]^T$ mobil robotun ağırlık merkezine uygulanan güç vektörüdür. M robotun ağırlığıdır. ϕ sabit eksen sistemi ile hareketli eksen sistemi arasındaki açıyı gösterir.

$$\overset{w}{m}R = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{EK-4.2}$$

$$\dot{S}_w = \overset{w}{m}R\dot{s}_w \tag{EK-4.3}$$

$$F_w = \overset{w}{m}Rf_m \tag{EK-4.4}$$

$f_m = [f_x \ f_y]^T$ hareketli eksene uygulanan güç vektörüdür.

Daha Sonra Eşitlik EK-4.1'den eşitlik EK-4.5 elde edilir.

$$M\left(\overset{w}{m}R\dot{R}_m\dot{s}_m + \ddot{S}_m\right) = f_m \tag{EK-4.5}$$

Mobil robotun dinamik özelliklerinden aşağıdakiler tanımlanabilir:

$$M\left(\ddot{x}_m - \dot{y}_m\dot{\phi}\right) = f_x \tag{EK-4.6}$$

$$M\left(\ddot{x}_m - \dot{y}_m\dot{\phi}\right) = f_y \tag{EK-4.7}$$

$$I_w\ddot{\phi} = M_1 \tag{EK-4.8}$$

Burada I_w robotun atalet momenti, M_1 robotun momentini gösterir ve f_x, f_y, M_1 :

$$f_x = -\frac{1}{2}D_1 - \frac{1}{2}D_1 + D_1 \quad \text{(EK-4.9)}$$

$$f_y = \frac{\sqrt{3}}{2}D_1 - \frac{\sqrt{3}}{2}D_2 \quad \text{(EK-4.10)}$$

$$M_ı = (D_1 + D_2 + D_3)L \quad \text{(EK-4.11)}$$

Ayrıca sistemin ters kinematik denklemlerinden:

$$r\dot{q}_1 = -\frac{1}{2}\dot{x}_m + \frac{\sqrt{3}}{2}\dot{y}_m + L\dot{\phi} \quad \text{(EK-4.12)}$$

$$r\dot{q}_2 = -\frac{1}{2}\dot{x}_m - \frac{\sqrt{3}}{2}\dot{y}_m + L\dot{\phi} \quad \text{(EK-4.13)}$$

$$r\dot{q}_3 = \dot{x}_m + L\dot{\phi} \quad \text{(EK-4.14)}$$

Buna ek olarak sistemin motor modelinden:

$$I_w\ddot{q}_ı + c\dot{q}_i = ku_i - rD_i \qquad i = 1,2,3 \quad \text{(EK-4.15)}$$

$$D_i = \frac{1}{r}(ku_i - I_w\ddot{q}_ı - c\dot{q}_i) \quad \text{(EK-4.16)}$$

$$D_1 = \frac{1}{r}(ku_1 - I_w\ddot{q}_1 - c\dot{q}_1) \quad \text{(EK-4.17)}$$

$$D_2 = \frac{1}{r}(ku_2 - I_w\ddot{q}_2 - c\dot{q}_2) \quad \text{(EK-4.18)}$$

$$D_3 = \frac{1}{r}(ku_3 - I_w\ddot{q}_3 - c\dot{q}_3) \quad \text{(EK-4.19)}$$

Burada L tekerleklerin robotun ağırlık merkezine olan mesafesi, D_i her bir tekerin sürüş gücü, r tekerleklerin yarıçapı, I_w tekerleklerin atalet momenti, $\dot{q}_i$ tekerleklerin açısal hızı, k sürüş kazanç faktörü, u_i sürüş torkudur.

Böylece EK-4.6−EK-4.19 eşitlikleri kullanılarak eşitlik EK-4.20, EK-4.21 ve EK-4.22 elde edilir.

$$\ddot{x}_m = a_1\dot{x}_m + a_2^{'}\dot{y}_m\dot{\phi} - b(u_1 + u_2 - 2u_3) \quad \text{(EK-4.20)}$$

$$\ddot{x}_m = a_1\dot{x}_m - a_2^{'}\dot{x}_m\dot{\phi} - \sqrt{3}b(u_1 - u_2) \quad \text{(EK-4.21)}$$

$$\ddot{\phi} = a_3\dot{\phi} - b_2(u_1 + u_2 - 2u_3) \quad \text{(EK-4.22)}$$

Burada $a_1,\ a_3,\ b_1,\ b_2,\ a_2^{'}$ katsayıları yukarıdaki denklemeleri kullanarak aşağıdaki gibi elde edilir.

$$f_x = -\frac{1}{2}D_1 - \frac{1}{2}D_1 + D_1 \quad \text{(EK-4.23)}$$

$$f_x = -\frac{1}{2r}(ku_1 - I_w\ddot{q}_1 - c\dot{q}_1) - \frac{1}{2r}(ku_2 - I_w\ddot{q}_2 - c\dot{q}_2) + \frac{1}{r}(ku_3 - I_w\ddot{q}_3 - c\dot{q}_3) \quad \text{(EK-4.24)}$$

$$f_x = -\frac{1}{2r}[(2ku_1 - ku_2 + ku_3) - (I_w\ddot{q}_2 + I_w\ddot{q}_3 + 2I_w\ddot{q}_3) + (c\dot{q}_3 + c\dot{q}_2 + 2c\dot{q}_1)] \quad \text{(EK-4.25)}$$

$$M(\ddot{x}_m - \dot{y}_m\dot{\phi}) = \frac{k}{2r}(u_1 + u_2 - 2u_3) - \frac{I_w}{2r}(\ddot{q}_1 + \ddot{q}_2 - 2\ddot{q}_2) + \frac{c}{2r}(\dot{q}_1 + \dot{q}_2 - 2\dot{q}_3) \quad \text{(EK-4.26)}$$

Eşitlik EK-4.12, EK-4.13 ve EK-4.14'ten;

$\ddot{q}_1 + \ddot{q}_2 - 2\ddot{q}_2 = -\frac{3\ddot{x}_m}{2r}$ ve $\dot{q}_1 + \dot{q}_2 - 2\dot{q}_3 = -\frac{3\dot{x}_m}{2r}$ eşitlikleri elde edilir.

$$M(\ddot{x}_m - \dot{y}_m\dot{\phi}) = \frac{k}{2r}(u_1 + u_2 - 2u_3) + \frac{3I_w\ddot{x}_m}{2r^2} - \frac{3\dot{x}_m}{2r^2} \quad \text{(EK-4.27)}$$

$$\ddot{x}_m = \frac{kr}{2Mr^2 + 3I_w}(u_1 + u_2 - 2u_3) - \frac{3c}{2Mr^2 + 3I_w}\dot{x}_m + \frac{2Mr^2}{2Mr^2 + 3I_w}\dot{y}_m\dot{\phi} \quad \text{(EK-4.28)}$$

f_y'de benzer şekilde bulunur,

$$f_y = \frac{\sqrt{3}}{2}D_1 - \frac{\sqrt{3}}{2}D_2 \quad \text{(EK-4.29)}$$

$$M\left(\ddot{x}_m - \dot{y}_m\dot{\phi}\right) = \frac{\sqrt{3}}{2}D_1 - \frac{\sqrt{3}}{2}D_2 \quad \text{(EK-4.30)}$$

$$M\left(\ddot{x}_m - \dot{y}_m\dot{\phi}\right) = = \frac{\sqrt{3}}{2}\left(\frac{1}{r}\left(ku_1 - I_w\ddot{q}_1 - c\dot{q}_1\right) - \frac{1}{r}\left(ku_2 - I_w\ddot{q}_2 - c\dot{q}_2\right)\right) \quad \text{(EK-4.31)}$$

$$M\left(\ddot{x}_m - \dot{y}_m\dot{\phi}\right) = \frac{\sqrt{3}}{2r}\left(\left(k(u_1 - u_2) - I_w(\ddot{q}_1 - \ddot{q}_2) - c(\dot{q}_1 - \dot{q}_2)\right)\right) \quad \text{(EK-4.32)}$$

$$M\left(\ddot{x}_m - \dot{y}_m\dot{\phi}\right) = \frac{\sqrt{3}}{2r}\left(\left(k(u_1 - u_2) - I_w\left(\frac{\sqrt{3}\,\ddot{y}_m}{r}\right) - c\left(\frac{\sqrt{3}\,\dot{y}_m}{r}\right)\right)\right) \quad \text{(EK-4.33)}$$

$$\ddot{y}_m = \frac{\sqrt{3}kr}{2Mr^2+3I_w}(u_1 - u_2) - \frac{3c}{2Mr^2+3I_w}\dot{y}_m - \frac{2Mr^2}{2Mr^2+3I_w}\dot{x}_m\dot{\phi} \quad \text{(EK-4.34)}$$

$$I_w\ddot{\phi} = M_1 \quad \text{(EK-4.35)}$$

$$I_w\ddot{\phi} = (D_1 + D_2 + D_3)L \quad \text{(EK-4.36)}$$

$$I_w\ddot{\phi} = \frac{L}{r}\left(k(u_1 + u_2 + u_3) - I_w(\ddot{q}_1 + \ddot{q}_2 + \ddot{q}_3) - c(\dot{q}_1 + \dot{q}_2 + \dot{q}_3)\right. \quad \text{(EK-4.37)}$$

$$I_w\ddot{\phi} = \frac{kL}{r}\left(k(u_1 + u_2 + u_3) - \frac{I_wL}{r}\frac{3L\ddot{\phi}}{r} - \frac{cL}{r}\frac{3L\ddot{\phi}}{r}\right. \quad \text{(EK-4.38)}$$

$$\ddot{\phi} = \frac{kLr}{3\,I_wL^2 + I_wr^2}\left(k(u_1 + u_2 + u_3) - \frac{3cL^2}{3\,I_wL^2 + I_wr^2}\dot{\phi}\right. \quad \text{(EK-4.39)}$$

Eşitlik EK-4.20, EK-4.21 ve EK-4.22 ile eşitlik EK-4.28, EK-4.34 ve EK-4.39 karşılaştırıldığından aşağıda gösterilen sabit değerler elde edilir.

$$a_1 = -\frac{3c}{2Mr^2+3I_w} \qquad b_1 = \frac{kr}{2Mr^2+3I_w}$$

$$a_3 = -\frac{3cL^2}{3\,I_wL^2 + I_wr^2} \qquad b_2 = \frac{kLr}{3\,I_wL^2 + I_wr^2}$$

$$a_2^{'} = -\frac{2Mr^2}{2Mr^2+3I_w}$$

ÖZGEÇMİŞ

Mustafa GEYİK

1978	Elazığ'da doğdu.
1993 – 1996	Elazığ Lisesinden mezun oldu.
1997 – 2002	Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünden mezun oldu.
2002 – …	T.C. Milli Eğitim Bakanlığında Bilgisayar Mühendisi olarak çalışmaya başladı. Elazığ il Milli Eğitim Müdürlüğü Bilgi İşlem Şubesinde Bilgisayar mühendisi olarak çalışmaya devam etmektedir.